AN AUGMENTED STEADY HAND SYSTEM FOR PRECISE MICROMANIPULATION

by

Rajesh Kumar

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

April, 2001

UMI Number: 3006288

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Steady Hand cooperative manipulation is a hands-on approach that integrates seamlessly in the surgical practice. In steady hand manipulation, the tool is held simultaneously by the user and the robot and the robot complies to forces applied by the user. Steady hand manipulation promises significant improvements in safety, accuracy over conventional practice at minimal cost and training to the user. It also offers a way around the difficult problem of encoding human intelligence, and preserves the benefits from experience and training.

We explore the possibility of encoding/utilizing task descriptions to improve transparency and performance of a steady hand manipulation task. This is done by constructing state space representations of the task. The user's interaction with the robot, tool-tissue interactions, and other sensory and planning inputs can be used to identify the task state and modify the behavior of the robot by using using optimized task and control parameters. Validation experiments for several cooperative tasks with and without augmentation are presented.

Advisor:
Dr. Russell H. Taylor,
Professor, Department of Computer Science

Acknowledgements

I consider myself fortunate to have been a member of the the Computer Integrated Surgery Lab and then the Engineering Research Center for Computer Assisted Surgical Systems and Technology. My association with the computer integrated surgery (CIS) research at Johns Hopkins has allowed me unlimited access to all aspects of CIS systems, providing a rich experience that is almost impossible to duplicate.

In particular, I would like to thank my advisor Dr. Russell H. Taylor for his support, encouragement patience, guidance and availability. His advice and helpful suggestions made this work possible.

My other advisors Greg Hager, Pat Jensen, and Louis Whitcomb were also sources of constant encouragement. I thank them for their help. My research also benefited from Dan Stoianovici's RCM module for the steady hand robot. I thank him for his help.

I was lucky to be surrounded by so many energetic and intelligent graduate students. Andy Bzostek helped me think through many research problems. And working with him has been a pleasure. Aaron Barnes built the steady hand robot that this work was based on. I also acknowledge his help with the experiments. I also thank Puneet Gupta, Jon Lazarus, Randy Goldberg, Adam Morris, Jianhua Yao, Alessandro Bettini, Samuel Lang and others for their help.

This work was funded in part by National Science Foundation, and in part by Johns Hopkins University. I thank them for their support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As robotic augmentation of surgical tasks role becomes common, its role is being compared with the role of automation in manufacturing. Just as automation helped improve efficiency, lower costs, and improve product quality in manufacturing, augmentation of surgery is predicted to reduce trauma, cost, time, and improve outcomes. This integration of technology is likely to lead to procedures being safer, more widely available, and to introduce new surgical procedures. These modern tools will also provide avenues for quick post-operative evaluation of each procedure, and statistical and comparative analysis of procedures by processing the data generated. Apart from providing better interfaces for training and evaluation of students, this will also allow iterative improvement (process learning), and quality management in surgical procedures.

The realization of this vision will require introduction of augmentation devices and systems in the operating room. These devices will integrate manipulation elements (e.g. active robots, passive braking systems, and other navigational aids), sensing elements (e.g. imagers, localizers and trackers), and visualization devices (e.g. displays, and projectors). They will provide high precision minimally invasive manipulation capability guided by multi-modality sensing and visualization. As the "intelligence" in these devices increases, they are likely to assume roles (what has been termed a "surgical assistant" [Taylor, 1999]) played by humans today.

However, as human integration of dexterity, perception, experience and judgment is currently irreplaceable these highly integrated systems are likely to be preceded by simpler systems performing only some functions. These devices are becoming common now. Robots are being used increasingly to improve accuracy, provide minimally invasive access, and as navigational aids. Examples of these systems include ROBODOC^{TM}([Paul *et al.*, 1992] [Taylor *et al.*, 1996, Taylor *et al.*, 1994]), NEUROMATE^{TM}(Integrated Surgical Systems, CA), MRT Robot (Brigham and Women's Hospital), JHU Steady Hand robot [Taylor *et al.*, 1999a]), JHU RCM/PAKY robot modules [Stoianovici *et al.*, 1998], JHU/IBM LARS System [Taylor *et al.*, 1996], Zeus^{TM}System (Computer Motion Inc, CA), da Vinci^{TM}(Intuitive Surgical Inc, CA), and PINPOINT^{TM}arm (Picker Gmbh.). They represent the next generation of surgical tools, and are not replacements for surgeons.

1.1 Surgical and Interventional Robotics

Robotic assistants are an attractive means of extending human capabilities and removing natural limitations. These limitations are most visible in fine, or dexterous manipulation tasks. Fine manipulation tasks involve high precision micrometer-level positioning accuracy, but only small ranges of manipulation forces and even smaller tool tip forces. Typically, these tasks will be performed by a human operator looking through a microscope while grasping a handle on the instrument or tool being used to perform the task. Performance of these tasks such as microsurgery is seriously affected by the limitations imposed by physical attributes of the sensory motor, muscular and skeletal systems on manual dexterity, precision and perception. These limitations affect the ability to hold an object steady (figure 1.1), the precision and smoothness with which a motion may be made and the scale of forces and textures which may be discerned [Gupta *et al.*, 1999, Cleeves & Findley, 1989, Bolles & Paul, 1973]. Natural factors such as physiologic tremor which occurs in all normal, active muscle, and drift (voluntary hunting and seeking motion of instruments about the desired point)

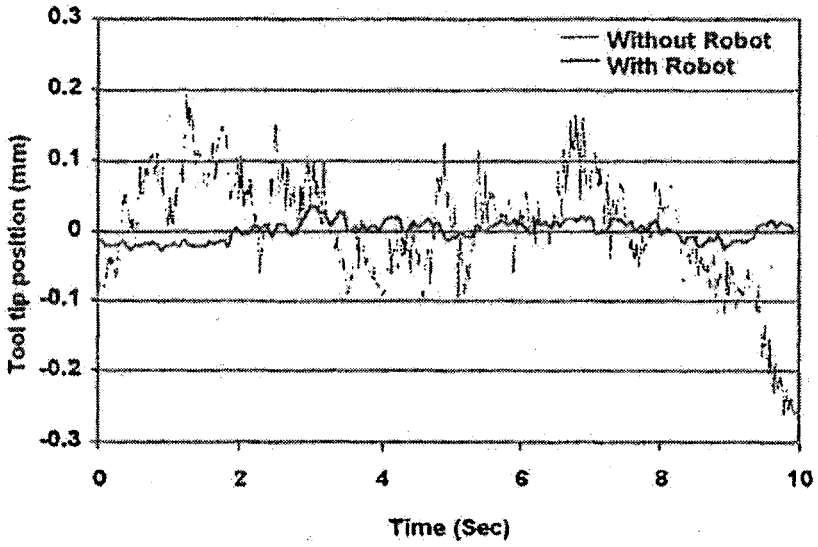

Figure 1.1: Tremor reduction by a cooperative paradigm. A common microsurgical tool is held by a user, and by a user and the steady hand robot together. Tremor is significantly reduced even though no filtering is used.

are some of the factors in establishing these limitations. Tremor further increases with stress, anxiety and following physical exertion. Robotic systems are inherently unaffected by these factors.

Most prior work on robotic micro-manipulation systems has emphasized traditional master-slave [Charles *et al.*, 1989, Hunter *et al.*, 1993] and tele-robotic manipulation [Jensen *et al.*, 1997]. These systems consist of two separate subsystems, a master manipulator at the task site, and a slave manipulator removed from the task site. The slave robot tries to faithfully reproduce the actions of the master, often scaled to suit the application. Combinations of position/velocity/force control are used depending upon the requirements of the task. Bilateral control approaches have also been explored.

The alternative, hand held or steady-hand[Taylor *et al.*, 1999a] manipulation, is a

hands-on approach. In steady hand manipulation, the tool is held jointly by human user and the robot. The robot is equipped with a sensor for user interaction and moves in compliance to user forces. It still acts a slave to user motions but because the user is manipulating the robot as he would the tool, there is no scaling of motion. The sensory input driving the motion can be scaled appropriately instead. Various position/velocity/force based controllers can be used depending upon the task requirements.

There are advantages and drawbacks of both approaches discussed above. The advantages of teleoperation approach include a better, more comfortable operating field for the surgeon, and scaling of sensory information, motion commands. The disadvantages include cost and complexity involved due to two manipulators, loss of kinesthetics and changes to existing practice, extra training etc.

The advocates of steady hand manipulation approach cite the simplicity, closeness to conventional procedures and therefore ease of use, reduced cost (due to a single manipulator), and improved kinesthetics among the advantages. The drawbacks include lack of ability to scale motion, and lack of remote operation.

For micro-manipulation, the benefits of steady hand far outweigh the disadvantages. The drawbacks are certainly important abilities but they are not crucial (or even not applicable) in micromanipulation tasks. Further, in applications like microsurgery, surgeon acceptance is crucial and approaches that do not require a complete re-engineering of the surgical workstation are much easier to introduce into practice.

1.2 Supervisory Control

In both approaches discussed above, the operator has direct control of the manipulator and its motion. This is desired, since autonomy of a robot is often seen as reducing safety by surgeons (e.g. What if the patient sneezes?). However, there are actions in any augmented task that are better performed by robots alone, or by robots under limited guidance from the user. This calls for a hybrid system, where the human maintains overall control, and monitors the execution of some portions of

the task performed by the robot autonomously.

The degree of autonomy given to the robotic system can be dynamic and decided by the human expert. This paradigm is referred to as supervisory control by Sheridan [Sheridan, 1988, Sheridan, 1998] and can form the basis of augmenting surgery. He describes supervisory control to mean one or more human operators setting initial conditions for, intermittently adjusting, and receiving information from a computer that itself closes a control loop in a well defined process through artificial sensor and effectors.

How does a supervisory control framework deal with augmented surgical tasks? Surgical tasks impose stringent requirements on an augmentation system. Goals for designing a good system include:

- Safety: includes identification of critical portions of the controlled task, ability to identify and/or correct faults, and redundancy to some extent. In medical procedures, the criticality of the task puts safety as the most important design consideration.
- Stability: performance meeting specifications over time, state/condition and over the range of inputs possible
- Efficacy/Accuracy/Functionality: ability to perform useful function identified by users, and to perform the function without significantly modifying existing processes.
- Ease of Interaction: ability to interact with the user with conventional tools used in the process and without imposing significant training or restrictions on existing practice.

Interaction with the planning process, possibility of learning/teaching, and accounting/process learning are other desirable attributes. It is difficult to design an optimal solution for tasks in such a dynamic environment and the flexibility of the system to allow tuning of its performance is also important.

1.3 Problem Statement

Steady hand manipulation is very well suited for supervisory control. Just as in conventional manipulation, the tool held by the manipulator performs a single function at any time, and the sequence of these actions defines the surgical task. However implementation of other requirements of supervisory control requires augmentation of compliant control with additional sensory information.
The core issues in this problem are

1. understanding the motions being made (or specifying them in some way)
2. designing task strategies and control strategies for efficient augmentation,
3. integrating non-sensory information (planning information, constraints etc.),
4. evaluating human performance with and without augmentation.

Each of these problems is large in its general form and beyond the scope of this work. A more time bound and tractable formulation of the problem is the following:

> *Generate a set of generic robust primitives, and ways to combine them to transparently implement explicit specifications for simple manipulation tasks.*

This reduces the core issues to:

1. creating robust primitives for common motions (and ways to compose them),
2. *implementing* control strategies for augmented tasks,
3. integrating *explicit representations* of non-sensory information,
4. evaluating human performance with and without augmentation for *implemented tasks.*

The domain of this problem is motions made by a surgical robot under direct surgeon guidance. A subset of motions made during surgery are similar to assembly analogues, such as positioning a tool to the surgical site (to the peg in hole problem) thus some inferences could be drawn from work in assembly environments. However, due to complex interactions with the environment and increased safety requirements only limited help is available from assembly tasks where the problem may be a little easier due to limited cost of an error and constrained environments.

Further, we are primarily interested in manipulation tasks with a degree of contact compliance between the tool and the environment being manipulated. Usually, these are relatively low information bandwidth tasks with moderately small force ranges. The environment interaction (i.e. tool tip) forces require special instrumentation, but this is available as part of the experimental platform.

We develop a system for executing explicit task level specifications of microsugical tasks, and validate them with experiments. Section 2.3 lists the contributions from this work in greater detail.

1.4 Organization of this document

This section provides an introduction to interventional robotics and discusses the importance of task level augmentation in computer-integrated surgery. A supervisory approach to task level augmentation is discussed and finally a workable problem statement is presented.

The **overview** section provides context to the problem of task level augmentation. A large body of research is available for task level programming and this is briefly outlined. The tasks under consideration are broadly classified into groups. This section also provides a basic definitions and overview of our approach. Finally we point out the contributions from this work and its applications.

The **system description** section details the organization and parts of the experimental environment. Both hardware and software components are described. The steady hand robot on which the experimental platform is based is described in de-

tail. The software controlling the robot is overviewed. A simple controller and its implementation are discussed.

Our **task architecture** is described in the next section. A complete description of the task graphs is provided. We discuss advantages of using composition of simpler skeletons to represent these tasks and compare them with other possible representations. We discuss task graph construction and usage with example tasks.

The **experiments** section discusses several example tasks. The experimental protocol is presented. The experimental setup, and the task requirements for each task are discussed. Performance parameters are identified and means of measuring them are discussed. Task graphs are constructed for these tasks, and analyzed. Results of each experiment are presented and discussed.

The **conclusions** section presents a summary of the work and provides a discussion on the results. Finally we include the scope and possible extensions to this work.

Chapter 2

Overview

Recently, sophisticated systems have been developed to assist, and augment human actions in medicine. However, they are often used to replace or reduce the number of people involved in a task (i.e. to act as tool or camera holder or perform a similar routine task) more than for their superior manipulation abilities. This is similar to the industrial environment where most of the initial automation consisted of teleoperated devices handling simple tasks or reproduced explicitly taught motions repeatedly. It has been realized that precision, efficiency, and consistency of human performance limit the current practice in several fields of surgery. This is very evident in fields that require high dexterity, long durations, or high precision of motion. But due to the complex nature of surgical manipulation and difficulty in modeling the environment augmented systems perform few augmented, automated or "intelligent" functions.

Most prior work on task level automation research has used structured assembly environments for experiments. This has partly been due to lack of availability of an experimental paradigm and platform suitable for unstructured environments. Our steady hand system provides us such a platform. The system meets the requirements of simple fine manipulation tasks common in surgery. This allows us to apply and extend the research in task level automation to surgical environments.

Most microsurgical tasks involve hand held tools operated under visual control.

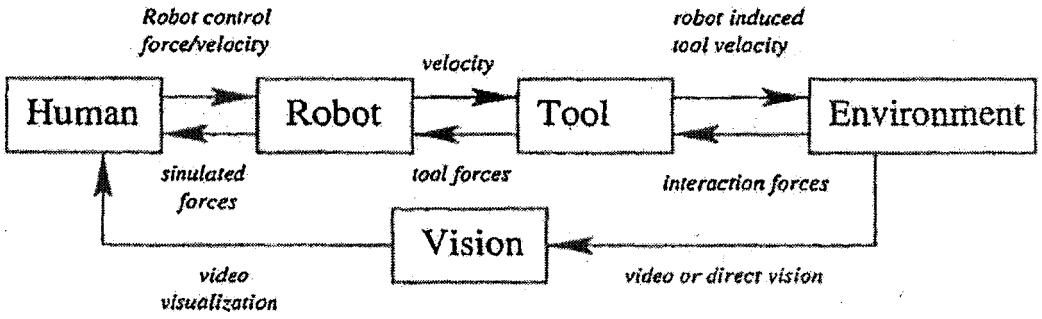

Figure 2.1: A task model of augmented tasks for a cooperative paradigm. The user interacts with the robot, by imparting forces on the tool, the tool in turn interacts with the environment. The robot simulates forces for the user based on forces imparted by the user, and those sensed from the environment. A visual channel (commonly a high-power microscope for microsurgery) is used to provide vision, and other augmented visualization.

In gross manipulation tasks forces from the tools are available, whereas fine manipulation is almost completely visual [Gupta *et al.*, 1999]. The user imparts forces on the tool causing it to move and senses the reaction (visual and forces) due to interaction with the environment. While Visual feedback maintains stability and improves performance, it is particularly important in fine and dexterous tasks and in minimally invasive tasks where there is little or no force feedback. Surgical practice emphasizes training, and kinesthetics of a surgical procedure is extremely important to the performance of a trained user.

Teleoperated task execution involves separation of the user from the work site. Some teleoperated systems seek to overcome this shortcoming by co-locating the slave and the master [Charles *et al.*, 1989], but significant kinesthetics and training issues remain unresolved. On the other hand, a steady hand task involves manipulating the shared tool operation under visual control. The user still imparts forces on the tool and causes the robot to respond accordingly. Augmented (figure 2.1), steady hand provides force feedback for all tasks, reducing the difficulty of fine tasks by restoring tactile feedback. Augmentation from other sensors enhances safety of the operation.

2.1 Background

This work relates to many areas of research in robotics and computer integrated surgery. This includes surgical manipulator systems, manipulation paradigms, high-level and low-level robot control, and representation, analysis and execution of augmented and automated tasks.

Surgical Robots

Surgical robotic systems relevant to this work include a master-slave system developed by Hunter [Hunter *et al.*, 1993], based on parallel manipulators. It was designed for experiments in eye surgery, manipulation of micro scale objects in surgery and tissue mechanical testing. The system has bi-directional force and motion feedback. Force scaling and motion scaling were also implemented. Audio and visual information was also exchanged between the slave and the master system with the surgeon using a helmet to control a stereo camera system.

The JPL Robot Assisted Microsurgery (RAMS) [Schenker *et al.*, 1995] master and slave system was a very sophisticated system designed for microsurgery. This system consisted of 6 degrees of freedom position scaling micro-telemanipulator with bi-directional force reflection and some tremor compensation.

Misuishi et al [Misuishi *et al.*, 1997] developed a master-slave system consisting of a vision system, a remote centered slave manipulator, and a master manipulator without force feedback. This system was designed for micro-drilling and micro-sawing experiments.

Another teleoperated parallel manipulator with micro-precision was designed by Grace et al[Jensen *et al.*, 1997]. This system was controlled by an open loop master controlling the position-velocity served slave. The goal of this joystick operated manipulator was to perform retinal vein annulations.

A prototype master-slave system was developed by Kwon et al. [Woo *et al.*, 1998] composed of a 6 DOF parallel slave micromanipulator and a 6DOF parallel haptic master manipulator. The slave robot is composed of a commercial industrial robot for

macro motion, and a micro-positioning robot equipped with interchangeable surgical instruments. Both master and slave are co-located in the operating room.

Some surgical robotic systems, though not designed for micro-scale operations are relevant as they have use some level of task information for constrained control. Davies [Davies *et al.*, 1997b] designed 3 DOF and 4 DOF manipulators with cutter end-effectors for orthopedic surgery applications. A PC is used for control in conjunction with a motion controller. The motion controller independently controls the motors. This system uses motion constraints to enhance safety. Constraints are stored as a lookup table to restrict compliant motion. Troccaz [Troccaz & Delnondedieu, 1996] designed a passive breaking system, that allows dynamic constraints to be applied on the users workspace. The system disables or enables motion dynamically thereby imposing geometric constraints on the workspace.

Other robotic systems

Industrial robot system have also been designed for applications at micro-scale. One of the leading goals for such systems is the assembly of micro electro mechanical systems (MEMS) devices [Codourey *et al.*, 1997, Muller *et al.*, 1996].

Manipulator Control

Control of robot systems operating in tandem with humans has been an active area of research for several decades. Most of the previous work has been in the industrial environments or using industrial manipulators. The equivalent of a power steering has been proposed for robots by Kazerooni et al [Kazerooni, 1989a, Kazerooni, 1989b] for applications to exoskeletons worn by humans [Kazerooni & Jenhwa, 1993].

A large body of literature is available on compliant control. This includes work on impedance control (e.g. [Hogan, 1985], [Whitney, 1977]), and stiffness control (e.g. [Salisbury, 1980]). Mason [Mason, 1982] formalized the use of geometric constraints in compliance specifications.

Kosuge et al [Kosuge *et al.*, 1993b, Kosuge *et al.*, 1993a, Kosuge & Kazamura, 1997,

Kosuge *et al.*, 1993c] have also looked at executing cooperative tasks, and extracting control information by using vision techniques.

Guarded Moves

Robotic systems have been applied to automation of industrial assembly tasks for a longer period and consequently the development of efficient user interaction methods has been a focus of research. The earliest work in performing a task as a sequence of steps is perhaps by Paul [Bolles & Paul, 1973], who introduced the notion of a guarded move and applied it to produce a centering grasp primitive. Guarded moves were motions that terminated on force or position thresholds. Bolles and Paul presented an application of guarded moves for assembling a model T Ford water pump. Branching control was used to deal with errors. Extra sensing was provided by an early vision system and free uncontrolled joints were used for compliance. Guarded moves have their limitations as they are limits on a single sensor input, and therefore not powerful enough to deal with an unstructured environment, where a combination of sensors may cause an undesirable state.

Manipulator Programming

Several authors have investigated planning sequences and high-level manipulator programming for automated assembly tasks. Early systems include [Taylor, 1976, Lozano-Pérez, 1976, Sanderson *et al.*, 1990] The work of Taylor [Taylor, 1976] is particularly relevant to this work because he introduced the specification of task skeletons built upon a library of primitives [Taylor, 1976]. These skeletons provided nominal values for the control parameters. The actual values were updated during the execution. Taylor also investigated numerical propagation of error and uncertainty through these skeletons. The skeletons provided the strategy for the task, and were written explicitly in the syntax of the language used (e.g. the Stanford AI manipulator programming system [Finkel *et al.*, 1975]).

Flexible Automation Systems And Task-Level Control

Flexible automation has a wide body of relevant work. Taylor et al [Taylor *et al.*, 1985] proposed a general purpose architecture for programmable automation research that uses task level programming. In this system, a multi-processor real time system performs all critical functions. The real time system shares some memory space with an interactive programming system. The programming system executes programs in an enhanced version of AML [Taylor *et al.*, 1982], talking to the real-time system by sending it high-level commands or *verbs*. The *verbs* may be simple motion specifications or termination conditions or compound *reflex graphs* composed of the verbs. The entire process is specified as *data flow graphs* using computational functions and shared state variables.

The user interacts with this system using a menu and pendant driven interface, common to most manufacturing applications. These interfaces drive an underlying programming language layer. The programming language allows specification and composition of *verbs* required. An underlying layer also allows extension of the system by constructing new *verbs* from real-time subroutines that perform device i/o, control law, trajectory planning and such.

A verb process in this system may consist of a number of cooperating real time tasks for various parts such as path interpolation, kinematics, dynamics and such. These are defined in terms of data flow graphs on the real-time subroutines and the shared state variables. The execution of a program is based on an *action sequence*, a list of subroutines generated by processing the data flow graph. This sequence is executed strictly sequentially on a single processor. Taylor envisions interrupts triggering the execution of a sequence.

Another system, proposed by Sanderson and Perry [Sanderson & Perry, 1983] describes multiple levels of descriptions for an assembly task. The highest level is an assembly algorithm, a sequence of operations performed on a physical part. This algorithm is described in a configuration-specific implementation using a sensing-representation-manipulation (SRM) description. A SRM description describes the

system elements, as well as task description. The environment is also described in form of both physical representation mapping sensor and control information to cartesian levels, as well as feature space representations mapping control and sensor information to sensed features. The task representation consists of both the This high level description is then translated to machine-level description for execution.

An implementation of this system also consists of multiple processors controlling individual manipulators, and sharing information with a supervisory system over ethernet network. A simple implementation with a single computer controlling a XY table, a vision system, and a manipulator and interacting with a supervisory system is also envisioned. The supervisory system interacts with a CAD design system to obtain information about the part to be assembled.

Taylor [Taylor, 1976] also analyzed automatic generation and analysis of task sequences, and task level planning in position space. Mason [Mason, 1982, Mason, 1978] analyses robot control strategies, and the effects of constraints on such planning in velocity space.

Other Approaches

Brooks put forward the idea of using behaviors for robot control. Behaviors are feedback loops connecting sensors to manipulation [Brooks, 1985, Brooks, 1990]. Behavior approaches seek to exploit context specific controllers to improve performance. Such an approach might use multiple controllers to perform various portions of a task, instead of a single controller.

Recent work in task-level control includes feature based programming, where a nominal path is used to plan, ignoring any uncertainty. The robot is commanded as if there was no error, and a set of local feedback controllers is used to keep the robot on its path. Eberman [Eberman, 1995] extends this idea by using sensor measurements to compute the location of the robot in the task configuration space. A search through the task graph provides this location, and also a decision process by which an event driven system could operate. On the other hand, Pook [Pook, 1995] looked at diectic

strategies for specification of events. She used a simple sign language to provide the context for execution of autonomous routines, thereby avoiding the search problem completely.

2.2 Our Approach

Finite state transducers and discrete event systems have been common in several areas, for example manufacturing systems, robotics and such unrelated areas as linguistics, natural language processing. Supervisory control of discrete event systems [Cassandras *et al.*, 1995] is also an area of research. Ramadge and Wonham [Ramadge & Wonham, 1989] give a broad overview of automata and formal language models for discrete event systems.

Event driven task level programming has often modeled tasks as a graph connecting discrete control strategies. The transitions between these control strategies were driven by measurements from sensors. Simple techniques [Howe & Cutkosky, 1991] have used collections of thresholds as a decision processes. More complex approaches [Hannaford & Lee, 1991, Eberman, 1995] use more powerful statistical tools to drive the transitions. All of these approaches, stress sensing from the environment as the driving force for the algorithm. By contrast, in cooperative manipulation the major driving signals are the forces applied to the tools and other discrete events created by the human user.

Cooperative tasks involving hand-held tools can be abstracted as controlled interaction of the tool/end-effector with the environment. Further, in steady-hand manipulation each of the manipulation steps is a compliant motion (or some other basic control primitive) with some termination predicate. The sequencing of these steps depends upon human control and effects of the interaction. There can be multiple outcomes in any given interaction and therefore multiple state transitions from the current state.

The idea is to impose a set of decision processes on the basic closed loop control. These decision processes use sensing and measurement to modify the control strategy

and effect state transitions. The decision processes are generated from an explicit task representation provided by the user for a specific task including planning inputs, safety considerations, and performance parameters. The system generates a representation for the corresponding finite state machine (FSM). Each state of this FSM represents a step in the task, and each transition a termination predicate. We detail the composition of states and transitions in detail below.

A standard approach in task level specification for manipulation has been to use well-defined implemented sensing and manipulation subroutines as the basis for composition. These subroutines (or just routines from here) form the building blocks of the system, and are coded in a programming language of choice. They perform basic sensing and manipulation, planning, or accounting functions. Manipulation subroutines such as move to specified position, move relative distance, move with specified velocity, and sensing subroutines such as get current value, filter raw values, bias values, and resolve values with respect to a given Cartesian frame are examples of routines.

Routines are the vocabulary for creation of states and transitions. They are used to compose primitives. Primitives are constructed by choosing inputs, and nominal parameters for sensing and manipulation routines. The primary manipulation primitives in steady hand manipulation are those that support compliant motion. A "move in compliance to forces" primitive is composed of a sensing routine to obtain resolve value of a sensor, a control law (e.g. velocity proportional to force error), and manipulation routine that receive the output of the control law. Geometric information can be passed using planning primitives, and safety primitives can be used to monitor abnormal conditions. Sensing predicates include both automatic sensing (e.g. contact detection) and explicit user input (e.g. input from buttons or a foot pedal).

Actions are distinctly identifiable portions of a task. An action may involve one or more manipulation routines. Events form the conditions that must be met before the action is executed and guards that cause the manipulation primitive to yield control.

The user provides explicit representation of the task as a sequence of actions. This is essentially a task skeleton. It is composed of actions (e.g. for peg in hole, actions

for positioning over the hole, inserting, and detecting contact etc.). The parameters for the routines are specified nominally by the user and appropriate parameters are then generated by the system for a particular task.

Our approach relies on the assumption that one skeleton may be sufficient for specifying a task strategy. Given that cooperative manipulation imposes a sequential ordering on planned task execution (user only performs one manipulation action at a time with the manipulator), a task skeleton is a simple and sufficient mechanism for representing task strategies. The skeletons need to be robust to geometric and environmental changes (some of which are expected and specified by the user as planning primitives) for this to be true.

The skeleton sequence is parsed and used to construct the task graph. The task graph is a finite state machine, with action completion predicates deciding state transition. For example, a simple surgical task of placing a micropipette in a blocked vessel in the eye is composed of several actions: a) positioning the tool at the port, b) orienting it such that it can be inserted, c) insertion of the tool, d) adjusting the orientation of the tool for placement at site viewing through the visual feedback device (usually a high power microscope), e) approaching the site, and f) achieving contact. This task has both coarse manipulation (positioning and orientation leading to the port) and fine manipulation inside the organ. Each of these parts can be implemented as an action. The conditions that need to be met before each action are identified. They form the predicates for that action. If the actions are composed of several primitives, the process of identifying primitives is repeated for the actions. Finally the user identifies safety requirements, such as limits on motion, sensory values etc for each action. A Task representation is generated using these actions. This serves as a skeleton for the task graph. The task graph is then executed in a training environment. During execution the user may identify redundant or additional actions, predicates that modify the task graph.

The System maintains a basic set of states, and predicates. These include initialization and cleanup, a manipulation set, data collection set, and safety and error checking predicates.

2.3 Contributions

Steady hand paradigm holds the promise of significant improvement in surgical practice. This work explores useful augmentation of the steady hand manipulation paradigm to include task level descriptions, and additional sensing and planning inputs.

- *Framework for task specification*: A simple framework was designed and implemented for using task-level specifications for surgical manipulation tasks. Task level descriptions are an attractive means of encoding task specific information in this paradigm. We investigate task level descriptions for simple manipulation tasks. The basis will be a limited set of robust but flexible primitives. These will be composed to create actions based on explicit user specifications. Actions create system states and transition predicates which are added to the existing set of states (safety, sensor input and simplest manipulation states). State transitions may be triggered by user input or sensory input due to interaction. Execution of tasks is sequential but state transitions are not sequential or synchronized. This creates a useful flexible system for performing simple manipulation tasks in environments with detectable interactions.

- *Modeling and analysis of suitable tasks*: Several simple surgical tasks were considered as experimental tasks. Suitable strategies for these tasks were identified and analyzed. Simple tasks involving tool positioning, or guiding, and more complex tasks involving a combination of simpler tasks were considered.

- *Comparative performance measurement*: Performance criteria were identified for each task. Some of the selected tasks were then performed in conventional modes and with a variety of augmentation modes. Comparison of these measures provides baseline data for evaluating the performance of augmentation.

A Modular Architecture for Robot Control: Design and Implementation

Abstraction of device interfaces is a standard paradigm in software engineering. We have used this approach for modeling our robots. A modular architecture was designed for controlling the steady hand robots. This architecture is also suitable for several other surgical robot applications (e.g. percutaneous therapy). This architecture was implemented in C++. This has resulted in a large library for robots and sensors. The programming interfaces can be easily extended include additional manipulators and allows high level control independent of the robotic platform actually implementing the commands. The modular robot control (MRC) structure abstracts interfaces in layers. The lowest layers wrap hardware implementations (vendor specific sensor implementations, servo control for robots etc). This functionality is then used by logical abstractions (joints for robots, sensor interface for all sensors). The higher layers contain functionality for Cartesian control, command scheduling, and network support. This layering allows new robots and new configurations of existing robots to be quickly supported, often requiring only a change in run-time loadable configuration. The library provides Cartesian control, forward and inverse kinematics, and joint control interfaces for our robots. It also includes implementations of force sensor, and digital and analog i/o device interfaces.

Chapter 3

System Description

Hands-on guiding of surgical robots has been used by several groups, for example, Davies knee surgery system [Davies *et al.*, 1997b, Davies *et al.*, 1997a], ROBODOC hip replacement surgery system [Paul *et al.*, 1992, Taylor *et al.*, 1994], and the JHU/IBM LARS system for endoscopic surgery [Taylor *et al.*, 1996]. And other surgical microsurgical mechanical systems of interest, though not employing hands-on guiding include those developed using telerobotic principles [Charles *et al.*, 1989, Hunter *et al.*, 1993], systems with force reflecting configurations [Charles *et al.*, 1989], and systems using passive input devices [Jensen *et al.*, 1997].

3.1 Hardware And Equipment

This work used a robot specifically designed for steady hand manipulation. This steady hand robot [Taylor *et al.*, 1999a] is a new modular test bed for microsurgical applications and is ideally suited to meet the task requirements. Simple stable control algorithms for this robot exist for most tasks, and more complex control and end-effector instrumentation [Gupta *et al.*, 1999, Berkelman *et al.*, 2000] is subject of independent research. Further the robot has a well-developed software foundation. The robot was first reported in [Taylor *et al.*, 1999a, Taylor *et al.*, 1999b], and a brief description and specifications appear below. Any specialized instrumentation needed

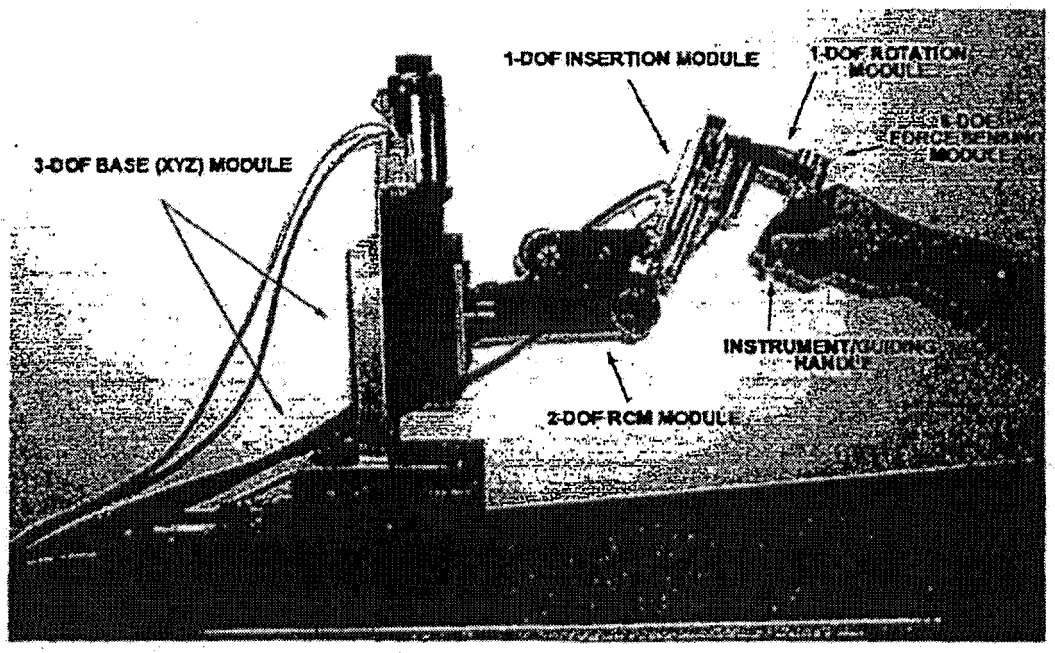

Figure 3.1: The "Steady Hand" manipulator

as part of the experimental apparatus is described where it was used.

The software used for this work was developed following standard paradigms, and some of it has proved useful for a wide range of unrelated research. The basis of the software system is a layered architecture of logical devices. These devices have abstract interfaces, and shield hardware specific details from the applications developed using them.

3.1.1 The JHU Steady Hand Robot

The steady hand robot is a 7-degree-of-freedom manipulator with XYZ translation at the base for coarse positioning, two rotational degrees of freedom at the shoulder (the RCM linkage [Stoianovici *et al.*, 1998]), an instrument insertion and rotation stages. A force sensor is built in the ergonomic end-effector. This robot has a remote

center of motion and an overall positional accuracy of less than 10 micrometers. A brief description of these module follows.

Base Translation Module

This three-axis base translation module is comprised of off-the-shelf motorized micrometer stages. The XYZ translation assembly is constructed by mounting a single axis Z-stage orthogonal to a X-Y table. Each axis has 100 mm of travel, can travel at speeds >40 mm/sec and has a positioning resolution ~2.5 μm (0.5 μm encoder resolution).

RCM: Remote-Center of Motion Module

The RCM module [Stoianovici *et al.*, 1998] is a novel compact and light robot for surgical applications that implements a fulcrum point located distal to the mechanism. It can precisely orient an end-effector (i.e. surgical instrument) in space while maintaining the location a point fixed in its workspace. For our purposes this module allows most tools to be oriented with a resolution of less then 0.01 degrees. RCM accommodates various end-effectors. This module has been successfully used at the Johns Hopkins Medical Institutions for several surgical procedures.

Rotation/Insertion End-Effector Module

The rotation and insertion end-effector[Barnes, 1999] provides linear displacement along the tool axis passing through the RCM point. The insertion stage has high stiffness, provides 150 mm of travel, can travel at speeds of ~ 30 mm/sec and has a positioning resolution of ~ 5-10 μm (1.5 μm encoder resolution). The rotation end-effector provides 360 degree continuous rotation about the tool axis and the mounting surface for the force sensor with guiding handle. It can travel at speeds of ~ 120-180 degrees/sec with a positioning resolution of ~ 0.05-0.10 degree (0.01 degree encoder resolution).

End-Effector Force-Sensing Module

This module uses a small commercially available force sensor (ATI Industrial Automation, NC) to capture user forces. The transducer has a resolution of 0.025N, 0.0625N-mm along the Z-axis and 0.0125N, 0.0625N-mm in the X-Y axes. Force ranges of ± 22.5N in the Z-axis and ± 12.5N in the X-Y axes can be measured. The torque range is ± 125N-mm. The force sensor is mounted on the instrument rotation stage with its z-axis parallel to the instrument insertion stage of the robot.

Tool Tip Force Sensors

Several tool tip force sensing devices are available. These include strain-gauge instrumented tools such as vireo retinal picks. Good single axis force sensing can be achieved by bonding a strain gauge pair in a half bridge configuration on a flat surface close to the tip of the microsurgical tool.

For prototyping purposes, small commercially available 3 axis force sensors (DX440 series, Bokam Engineering Inc.) are also available. These solid stainless steel sensors contain full bridge configurations, and provide X,Y,Z force sensing ability. The integrated signal conditioning electronics provides a linear proportional signal, that can be resolved to 10s mN with careful calibration.

Berkelman [Berkelman *et al.*, 2000] designed a new 3 axis force sensor for measuring tool tip forces. The force sensor has resolution ~0.05mN, a force range of 1N, and maximum range of 5.0N. The force sensor is based on a double cross design and is a cylinder 12.5mm in diameter and 12.25mm in height. The signal from the force sensor is amplified using a signal amplifier (Vishay Instruments Division, 2210A signal conditioning amplifier) with suitable gains and read using a 16 bit digital to analog converter.

3.1.2 Robot Control Hardware And Electronics

The robot hardware control runs on a Pentium-II 450MHz PC with Windows NT operating system. An 8-axis DSP series controller card (PCX/DSP, Motion Engi-

neering Inc, CA) is used to control the robot. The card provides servo control using a 40MHZ Analog Devices ADSP-2105 processor. It also has support for user digital and analog input, output lines. The PC also houses the ISA force sensor controller. The robot hardware control runs on a Pentium-II 450MHz PC with Windows NT operating system. A Matrox Meteor II video capture card is also housed in the PC to digitize the video signal from imaging devices.

3.2 Robot Models

The steady hand robot is a serial manipulator. Further, the mechanical configuration of the robot ensures that end-effector motions of the robot decouple about the remote center of motion point. This considerably simplifies modeling the kinematics of the robot.

3.3 Kinematics

The steady hand robot appears in figure 3.1. It uses only serial linkages. Both the forward and inverse kinematics of the robot can be computed in closed form.

3.3.1 Forward kinematics

The forward kinematics of the robot is easily computed. The kinematics equation of the steady hand robot is:

$$ {}^{f}T_{t} = {}^{f}T_{b} \cdot {}^{b}T_{a} \cdot {}^{a}T_{e} \cdot {}^{e}T_{t} \tag{3.1} $$

where f is the fixed frame, b is the base frame(at the end of the base translation module), a is the arm frame(at the end of the RCM module), e is the end-effector frame, and t is the tool frame.

The base frame of the robot is placed at the end of the Cartesian base translation module. It has three orthogonal translation stages, which give:

$$^{f}T_b = P_{x,d_0} \cdot P_{y,d_1} \cdot P_{z,d_2} \tag{3.2}$$

where P is a pure translation about the axis specified (i.e. a frame with identity rotation) and d_0, d_1, d_2 are the joint variables.

The arm frame contains the remote center of motion rotational module. This module consists of a rotation linkage about the X axis, followed by a constant offset about the Z axis, followed by a rotation about Y axis, and a constant offset about the Y axis. Using $R(i, \phi)$ to represent a rotation about axis i by angle ϕ, the arm frame can be computed as:

$$^{b}T_a = R_{x,\theta_3} \cdot R_{z,\phi} \cdot R_{y,\theta_4} \cdot R_{y,\psi} \cdot T_c \tag{3.3}$$

where θ_3, θ_4 are the joint variables. The constant offset rotations about the Z, Y axes ϕ, ψ are dependent upon configuration of the RCM module. T_c is a constant offset transformation that is identity for our end-effector when the RCM point is properly configured. The arm frame then is given by:

$$^{b}T_a = R_{x,\theta_3} \cdot R_{z,\phi} \cdot R_{y,\theta'_4} \tag{3.4}$$

where $\theta'_4 = \theta_4 + \psi$.

The two DOF end-effector module provides rotation and translation about the tool axis. The end-effector frame therefore is given by:

$$^{a}T_e = P_{z,d_5} \cdot R_{z,\theta_6} \tag{3.5}$$

where d_5, θ_6 are the joint variables.
This gives the complete forward kinematics as:

$$^{f}T_e = P_{x,d_0} \cdot P_{y,d_1} \cdot P_{z,d_2} \cdot R_{x,\theta_3} \cdot R_{z,\phi} \cdot R_{y,\theta'_4} \cdot P_{z,d_5} \cdot R_{z,\theta_6} \tag{3.6}$$

$$\begin{pmatrix} c_6c_4c_\phi + s_6s_\phi & c_4s_6c_\phi - c_6s_\phi & s_4c_\phi & d_0 + d_5(s_4c_\phi) \\ c_3c_4c_6s_\phi & c_3c_4s_6s_\phi & c_3s_4s_\phi - s_3c_4 & d_1 \\ +s_3s_4c_6 - c_3s_6c_\phi & +s_3s_4s_6 + c_3c_6c_\phi & & +d_5(c_3s_4s_\phi - s_3c_4) \\ s_3c_4c_6s_\phi & s_3c_4s_6s_\phi & s_3s_{4s\phi} + c_3c_4 & d_2 \\ -c_3s_4c_6 - s_3s_6c_\phi & -c_3s_4s_6 + s_3c_6c_\phi & & +d_5(s_3s_4s_\phi + c_3c_4) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.2: Closed form kinematics for the steady hand robot . c_i, s_i stand for $\cos(\theta_i)$, $\sin(\theta_i)$ respectively and d_i, θ_i are the joint variables. θ_4 is assumed to include the constant rotation offset about y after the RCM module, and ϕ is the constant offset after the rotation about x in the RCM module.

A given tool may have its own constant offset frame, and a tool frame is customarily added at the end:

$$^eT_t = Frame(Rotation(\theta_x, \theta_y, \theta_z)\ , Vec(t_x,\ t_y,\ t_z)) \tag{3.7}$$

where θ_i, t_i are constants offsets for a particular tool.

For a properly configured remote center of motion point closed form this can be computed to a matrix in figure 3.2 in closed form.

3.3.2 Inverse Kinematics

The steady hand robot has redundant degrees of freedom. However, the inverse kinematics can be computed in closed form given user by imposing constraints on the redundant degrees of freedom (instrument insertion, and gross positioning are rarely used together). Establishing a coordinate frame at the RCM point further simplifies computation.

Given a target frame $F = Frame(R(\theta), Vec(P))$ in the fixed coordinate system and

assuming user has applied constraints on the instrument insertion, the Cartesian positions are extracted as joint variables:

$$d_0 = P_x, \quad d_1 = P_y, \quad d_2 = P_z \tag{3.8}$$

where $Vec(P) = (P_x, P_y, P_z)$. Using the rotational part of forward kinematics, we obtain:

$$\theta_4 = sin^{-1} Rz_x / cos(\phi) \tag{3.9}$$

and using θ_4 we can compute $\alpha = s_4 s_\phi$, $\beta = c_4$

$$\frac{c_3 = \alpha Rz_y + \beta Rz_z}{\alpha^2 + \beta^2}$$

$$\frac{s_3 = \alpha Rz_z - \beta Rz_y}{\alpha^2 + \beta^2}$$

$$\theta_3 = atan2(s_3, c_3) \tag{3.10}$$

and similarly $\alpha = c_4 c_\phi$, $\beta = s_\phi$ gives

$$\frac{c_6 = \alpha Rx_x - \beta Ry_x}{\alpha^2 + \beta^2}$$

$$\frac{s_6 = \alpha Ry_x + \beta Rx_x}{\alpha^2 + \beta^2}$$

$$\theta_6 = atan2(s_6, c_6) \tag{3.11}$$

where the rotation matrix $R(\theta) = Rotation(\theta_x, \theta_y, \theta_z) = [R_x, R_y, R_z]$ and each of R_x, R_y, R_z are 3 vectors (x, y, z). The joint variable d_5 for instrument insertion remains unchanged. Similar other inverse kinematics Similar computations for other constrained cases can be performed. Since for most of the work the redundant degrees are not used together, these are very useful in practice.

3.3.3 Cartesian Control

For moving to a specified frame, Cartesian control simply involves performing inverse kinematics on the given frame and moving with obtained joint parameters. For computing the current Cartesian frame the process is the reverse. Obtain current joint parameters, and perform forward kinematics to get current frame.

3.3.4 Force Control: A Simple Controller

There is a large body of literature concerning provably stable control techniques for robots. The work most relevant to this is perhaps that of Kazerooni [Kazerooni, 1989a, Kazerooni, 1989b] for exoskeletons worn by a human operator [Kazerooni & Jenhwa, 1993] to amplify his strength. Kazerooni reports a linear systems analysis of the stability and robustness of the exoskeleton control systems where the manipulator magnifies the operator's forces by a scale of 10 or more. The stability analysis of such system is complicated by imperfect models for hydraulically actuated robots, and the operators arm. As a consequence it is necessary to accommodate variations in both human and robot arm dynamics.

We are interested in cooperative force control where the robot and the human manipulate a single tool in contact with a compliant environment. The steady hand paradigm *scales-down* the human operator's force input by a factor of 0.1 to 0.01. Further, micro-surgical manipulation tasks only require precision low-speed low-acceleration motion. These performance requirements allow us to use a simple position controlled manipulator model. The steady hand system described in section 3.1.1 is a compact and stiff manipulator actuated by highly geared electric actuators, attenuating arm dynamics terms. Each joint is individually controlled by a PID controller and joint velocities are limited for safety considerations. In normal operation, no unmodeled dynamic effects have been observed. So the plant model of a *position controlled manipulator* whose control is "desired position" is sufficiently accurate for present robot, performance requirements, and applications.

Below, we describe stable one-DOF force control law for the force control problem.

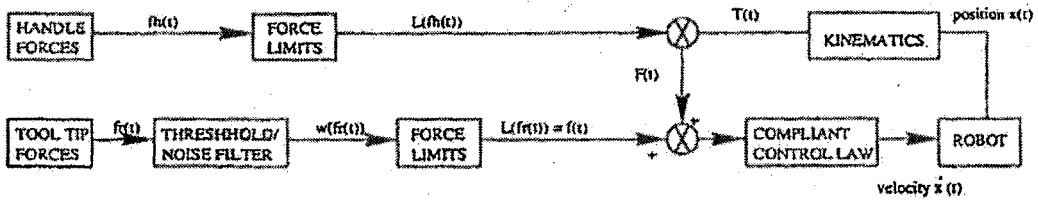

Figure 3.3: Block diagram of a simple force-proportional velocity controller

This approach was detailed in [Kumar *et al.*, 2000], and has also been extended for multi-DOF force control.

Consider the 1-DOF position controlled system in figure 3.4, with position $x(t)$, and velocity $\dot{x}(t)$. The force exerted by the tool tip on a compliant environment is $f(t)$N. The force exerted by a user on the rigid force handle on the robot is $F(t)$N. The robot's end effector and tool has a collective linear compliance of k_r N/m, and the compliant environment in contact with the tool tip has linear compliance k_eN/m. The tool and environment combination then has a compliance $1/k = 1/k_r + 1/k_e$.

The tool tip will be in equilibrium if $f(t) = kx(t)$. The system is position controlled, with the control input $x(t)$ (or $\dot{x}(t)$, where $\dot{x}(t)$ is the time derivative of $x(t)$) with the PID control for each axis ensuring the position and velocities track. So, $x(t) = x_d(t)$ and $\dot{x}(t) = \dot{x}_d(t)$.

where $x_d(t)$ and $\dot{x}_d(t)$ are the desired position and velocities respectively. Given a constant desired tool tip force $f_d(t)$, the force error $\delta f(t) = f(t) - f_d(t)$ converges asymptotically to zero, i.e. $\lim_{t\to 0} \delta f = 0$, and $\delta \dot{f}(t) = \dot{f}(t)$ since $f_d(t)$ is a constant.

It is easy to show that the control law $x_d(t) = -k_f \int_0^t \delta f(t) d(t)$ results in exponentially stable first order force dynamics of $\delta \dot{f}(t) = -k_f k \delta f(t)$.

A block diagram for this controller appears in figure 3.3. Force scaling can be accomplished by appropriately scaling down the force applied by the user force to obtain the desired force. More complex control laws are required for time varying forces, and when the robot or environment compliances are not known. Roy et al [Roy & Whitcomb, 2001] have worked on more sophisticated adaptive control strate-

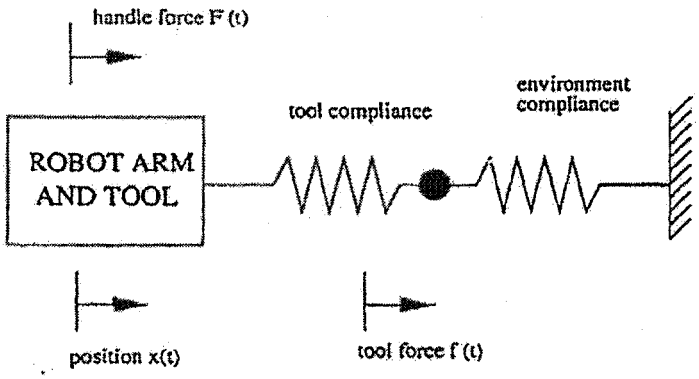

Figure 3.4: A 1-DOF position controlled manipulator in contact with compliant environment

gies, which could replace this controller if needed.

For our purposes, this simple control should suffice. This simple controller works as follows. User forces are sensed from the six degree of freedom force sensor attached axially to the handle held by the user. The axes of the tip force sensor are aligned with the end-effector frame of the robot. The axes of the handle force sensor are also aligned with the end-effector when the robot is in its home position. The force sensors are calibrated by computing a bias force with no external forces on the sensor. This bias force is stored in the robot base coordinate frame. During compliant motion, the system samples the current handle forces, and resolves them to the robot base frame. These resolved forces are offset by the bias and filtered using a simple high pass filter (W_f above). This biased signal is then amplified by gains k_f. The velocities computed are limited by user provided limits and the fed to the joints selected by the user (mode).

$fh(t)$: sampled force data from the handle sensor, a three tuple (fh_x, fh_y, fh_z),

$fr(t)$: sampled force data from the tip force sensor, also a three tuple (fr_x, fr_y, fr_z),

$F(t)$: forces resolved in the robot base frame,

$F_B(t)$: filtered and biased handle force values obtained from resolved forces,

$f_B(t)$: filtered and biased tip force values,

F_b: bias force for handle sensor(Fb_x, Fb_y, Fb_z) in the robot base frame,
f_b: bias force for the tip sensor,
$T(t)$: transformation from handle force sensor frame to robot base frame,
mode: Base X,Y,Z joints (1), or RCM rotation joints and insertion joints (2),
$k_v(mode)$: joint velocity proportional gains, based on user selected mode,
$\dot{x}(t)$: joint velocities for selected joints.

A windowing function for small errors:

$$W(f_x, f_y, f_z) = \begin{cases} < 0, f_y, f_z > & if -w_x < f_x < w_x, \\ < f_x, 0, f_z > & if -w_y < f_y < w_y, \\ < f_x, f_y, 0 > & if -w_z < f_z < w_z, \\ < f_x, f_y, f_z > & otherwise \end{cases}$$

A limiting function for joint velocities:

$$L_v(\dot{x}) = \begin{cases} \dot{x}, & if\ \mathrm{L}_l < \dot{x} < L_u \\ L_l, & \dot{x} \leq L_l \\ L_u; & \dot{x} \geq L_u \end{cases}$$

And a limiting function for forces:

$$L(f) = \begin{cases} f, & if\ \mathrm{F}_l < f < F_u \\ F_l, & F \leq F_l \\ F_u; & F \geq F_u \end{cases}$$

The controller can be formulated as:

$$\begin{aligned} F(t) &= T(t) \cdot L(fh(t)), \\ f(t) &= W(L(f\tau(t))), \\ F_B(t) &= F(t) - F_b, \\ f_B(t) &= f(t) - f_b, \\ \dot{x}(t) &= k_v(mode) \cdot (F_B(t) - f_B(t)), \text{ and} \\ \dot{x}(t) &= L_v(\dot{x}(t)). \end{aligned}$$

This controller has been implemented easily using hardware described above and performs satisfactorily.

3.4 Software Architecture

A library of C++ classes has been developed to implement the interfaces for our robots. This modular robot control (MRC) library provides Cartesian level control. The library is a set of portable C++ classes. It provides Cartesian level control for serial manipulators. It includes classes for kinematics, joint level control, command and command table management, sensor and peripheral support, network support. Some exception and error handling is also built in. An array of sensors including serial and parallel ports, ATI^{TM} force sensors, joysticks, digital buttons and foot pedals are supported.

Unrelated to this work, support is available for the proprietary LARS servo controller. Applications using this library can manage both local and remote hardware with minimum configuration and little coding. Some of the functionality is limited to WIN32 operating systems but most of the approach is platform independent and distributed. The library allows applications to use of any serial manipulator by adding minimal functionality to the joint level API and kinematics. Since most computer integrated surgery (CIS) applications are written while the hardware is yet to be constructed, it allows software development by using a simulator included in the library and easy replacement of the actual hardware API when the hardware is ready.

The MRC structure abstracts interfaces in layers (see figure 3.5). The lowest layers wrap hardware implementations (vendor specific sensor implementations, servo control for robots etc). This functionality is then used by logical abstractions (joints for robots, sensor interface for all sensors). The higher layers contain functionality for Cartesian control, command scheduling, and network support. This layering allows new robots and new configurations of existing robots to be quickly supported, often requiring only a change in run-time loadable configuration.

Control algorithms have been implemented based on the MRC library. For ex-

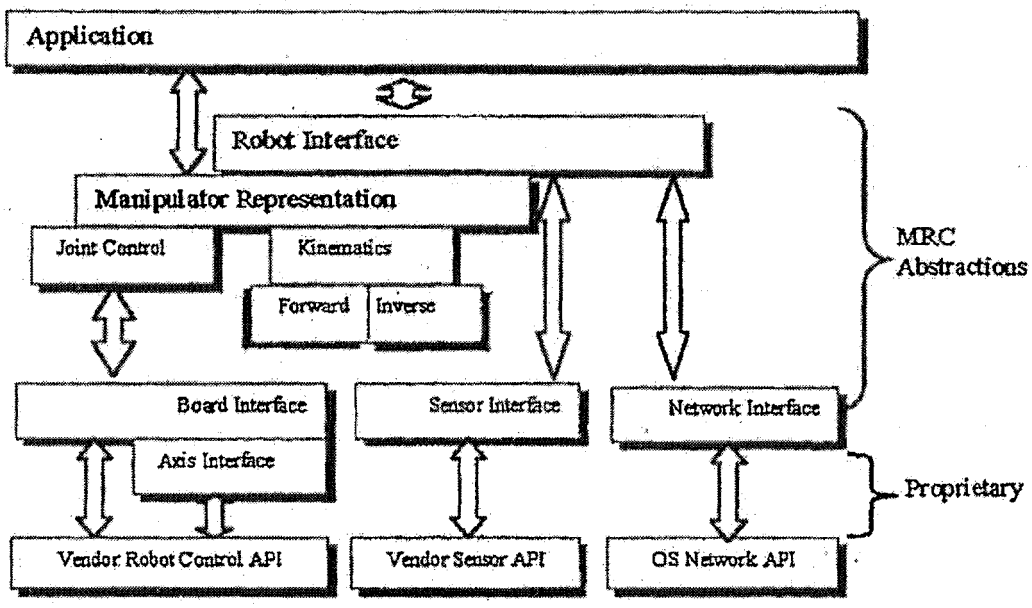

Figure 3.5: Hierarchical organization of robot control software

ample, for simple force proportional velocity control discussed above has been implemented using the MRC library. The base joints and the upper joints can be controlled independently by using a foot pedal. Control rates of over 1000Hz can be achieved using this controller.

3.4.1 Kinematics Issues

The kinematics interface consists of two parts forward kinematics, and inverse kinematics. For serial manipulators, D-H parameters provide a straightforward and simple means of constructing forward kinematics of the robot. We use a scheme based on D-H parameters to specify the forward kinematics of any robot. Even though we limit ourselves to only serial linkages, the inverse kinematics is better implemented explicitly for each configuration. This allows applications to specify preferences on joints where any redundant joints exist, and also limits on each individual module.

We are also looking at including simple geometric descriptions of the robot in the configuration file for visualization purposes.

3.4.2 Joint Control

The abstract joint control interface is based on the hardware specific board and axis layers (see figure 3.5). The board and axis layers abstract vendor-specific implementations. Depending upon the functionality of the motion control board being used (such has the MEI controller board) by the current configuration, additional functionality may be available to the joint control API (such as coordinated moves). The board and axis interfaces are constructed to take advantages of this. The joint interface wraps all commonly available pre-emptive and non-pre-emptive position, and velocity control.

Chapter 4

Task Architectures

The standard approach in task-level planning with robots has been to first construct an accurate representation of the world. Any tasks are then specified in terms of this world representation and sensing is used to update this representation to maintain its accuracy. The task to be executed is then typically partitioned into parts that can be executed/monitored simultaneously. Strategies for these parts are developed individually and means for coordinating between them are defined. Often these parts are prioritized to simply the process.

The above approach suffers from several drawbacks. It is difficult to maintain a complete and correct representation of the world at all times. This is due to difficulties in sensing, and interpreting sensed signals for any dynamic environment. Some approaches (e.g. [Brooks, 1985]) have avoided this problem by using only current or very recent information for decision purposes. However, it is very difficult to integrate planning information in such representations. This makes these approaches rather unsuitable for complex planned tasks.

It is a hard problem to decompose a task into several sub-parts that can be monitored and executed simultaneously. It is often difficult to identify portions that can be simultaneously executed. However, multi-step strategies have been common in task level assembly planning [Taylor, 1976, Lozano-Pérez *et al.*, 1984, Taylor *et al.*, 1985, Sanderson & Perry, 1983]. A peg in hole is analogous to assembly operations for

mating parts, and inserting screws to fasten parts together and is probably the most common task analyzed for task strategies. For example, a common strategy is to bring the peg in contact with the hole, then align the peg with the hole using a contact search strategy, before using an insertion motion to place it in the hole. The insertion motion could be made more robust to uncertainty by using a compliant insertion motion. Another common example is strategies for grasping objects.

Multi-step strategies have also been combined together with sensor driven frameworks for event based programming. Brooks' [Brooks, 1985] subsumption architecture is a good example. Grasping strategies by Howe and Cutkosky and other systems for assembly programming have used event driven systems for assembly.

Combination of robot supervisory control with event driven programming has also been an area of research. Most previous research has focused on autonomous strategies. Learning by demonstration by has been explored by several authors [Kosuge *et al.*, 1993b, Kang & Ikeuchi, 1997, Morrow & Khosla, 1997, Iba *et al.*, 1999]. But most robotic tasks in assembly are planned to be autonomous, and cooperative strategies are rarely needed. An exception is pick and place operations, where a robot could cooperate with a human in moving heavy loads, that a human alone can not move.

A combination of supervisory control and event driven programming becomes very attractive as the scale of tasks being considered is reduced to a micro-scale. It is even more attractive for complex, practical tasks for which autonomous strategies are unsuitable. Microsurgical tasks are a good example. While it is challenging for even a trained surgeon to remove an epiretinal membrane, the difficulty is reduced when the same task is executed as a constrained motion task augmented with a robot. Autonomous strategies for surgical manipulation are also harder to implement due to variations in anatomy, the dynamic nature of the task, and for reasons of safety. An augmented execution mode provides benefits of human experience as well as superior accuracy of the robot.

In single tool hand-held surgical tasks, the user makes only one motion at a time with the single tool. This provides a natural ordering in which parts of the task are

to be performed. Further, in augmented steady hand manipulation the tool is held jointly by the user and the robot. This provides a continuous user input for driving the system. Interpreting this comparatively low bandwidth force signal for detecting task state is a simpler problem. It can be made easier if the user provides additional inputs to guide transition between some parts of the task.

The problem then reduces from automating a task to augmenting it. Augmented tasks are easier to specify as well. Planned execution provides a skeleton for task execution, and actual parameters for task portions can be optimized by training.

Further, this allows for additional sensing and planning inputs to be incorporated in the task skeleton. The effect of these inputs is decided by the user explicitly during the task skeleton description, and can range from completely impeding motion (to prevent a specified limit being crossed), and reducing degrees of freedom for manipulation (constraint or virtual fixture) to magnifying response in certain areas of the workspace. This essentially involves binding "good" parameter values on the detected task context.

4.1 Task graphs

A task graph represents an actual task. States of the task graph, or actions are constructed by breaking down the task into parts where detectable discontinuities require a change in the behavior of the robot. For example, moving from a portion of the task that involved positioning the tool to that requires orienting the tool or from an interactive part to a small monitored automated portion. The execution of these steps is decided by human control and effects of the interaction. There can be multiple outcomes in any given interaction and therefore multiple state transitions from the current state. The task skeleton provides planned/allowed state transitions from each state. In each state, sensing and measurement modify the control strategy and effect state transitions. The explicit task representation provided by the user for a specific task includes inputs, safety considerations, and performance parameters.

In a steady hand cooperative task, the user applies forces on the tool held jointly by

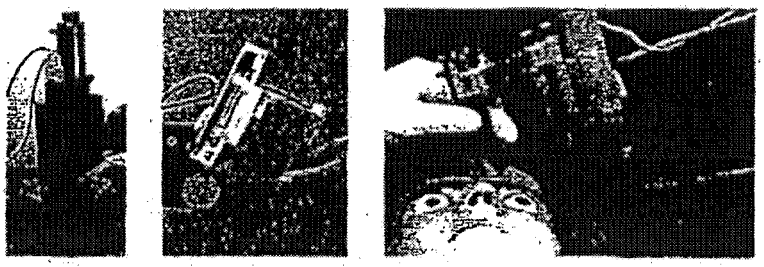

Figure 4.1: Common Manipulation modes for the Steady Hand Robot. The translational joints (left) in the base are used to position the tool in the workspace. The rotational joints, and the end-effector joints (center) are then used for dexterous manipulation. These modes are particularly useful, when the robot is used for a minimally invasive task(right), requiring that the remote center of motion be kept fixed at the port through which the tool is inserted.

the robot. The robot complies to the forces appropriately. The most basic cooperative task graph is one describing this steady hand manipulation. There are two distinct modes in which the steady hand robot is often used. The first (*translate*) is one that uses the translational base (see figure 4.1) to reposition the remote center of motion (RCM) point in the fine manipulation workspace. The second mode (*orient*) uses rotational and end-effector joints and is suitable for performing fine manipulation while keeping the RCM point fixed. The user may switch between the two modes, by using buttons on the foot pedal. The user signals completion of the task execution by pressing a third button on the pedal.

This simple three state task graph appears in figure 4.2. The two manipulation states in the task graph represent control strategies discussed above. The third (*done*) represents completion of task execution. There are several other modes of operation for the steady hand robot. The two modes of operation discussed above, involve three degrees of freedom each. Other modes may require more or fewer degrees of motion, such as one degree of freedom modes for insertion of a tool or rotation about the tool axis, to six degrees of freedom for simultaneously orienting and positioning the tool. In additional, when the forces applied by the environment on the tool tip are also used for control purposes, each of these modes has a variant where the scaled force

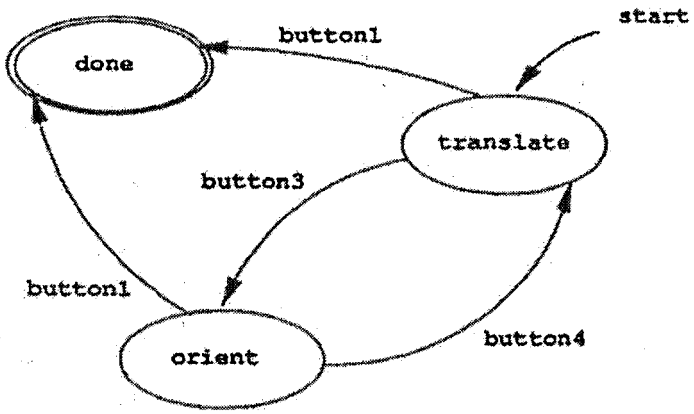

Figure 4.2: A simple task graph for control of the steady hand robot implementing basic control modes shown in figure 4.1. The user can chooses between the two modes by pressing buttons on a foot pedal.

error is used as the controlling input. The task graph in figure 4.2 can be modified to add actions for these modes.

The above task is navigated by the user by pressing a foot pedal. However, it is easy to imaging transitions being caused by interactions with the environment. For example, moving from a portion of the task with no contact to one in contact with the environment. Contact or impact can be reliably sensed automatically, and can be used for navigating the task.

4.2 Task Graph Construction

A simple model for these augmented tasks is a finite state machine(FSM) that contains a set of states corresponding to the actions for the task and the means to navigate these states. Pook [Pook, 1995] constructs similar finite state machines and uses discrete hand signs to direct the robot through the execution of such tasks as flipping eggs in a pan. She uses trained hidden markov models (HMM) to recognize the gestures made by the user. Here, we use discrete signals provided by the user such as buttons on a foot pedal, or simple thresholding of continuous signals to navigate

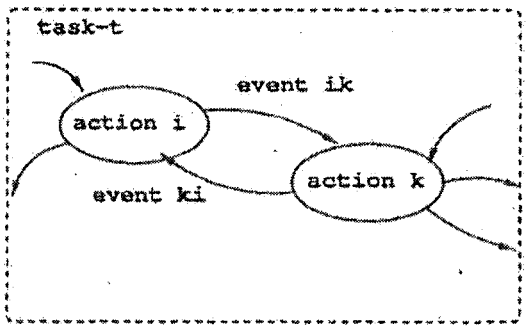

Figure 4.3: A finite state machine model for an augmented task similar to figure 4.2. Each distinct part of the task is a state of the FSM.

our task graphs.

The language described by the FSM (and consequently the task that can be specified using it) is restricted first by requiring that the primitives - actions (nodes in the state graph) and events (transitions) are composed entirely of a fixed grammar of *routines*. These routines are individually required to meet safety requirements of the tasks under consideration. Routines are the basic defined elements. They are used to compose primitives. Primitives are constructed by composing routines, and choosing inputs and nominal parameters for sensing and manipulation routines used.

Definition 4.1 (Routine) *An implementation of a sensing or manipulation function (with a known interface). These implementations perform clearly defined tasks. They form the lowest level of abstraction in the composition process.*

Routines operate on the manipulator, sensors, discrete input and outputs, imaging and planning information. Some Examples of commonly used routines are:

- manipulation routines -
  - Compliant motion routines: move in response to user forces. Several versions of these routines are possible depending upon the joint configuration selected,

- Constrained compliant motion routines: move in response to user forces, subject to applied constraints, for example ComplyToForcesInPlane, ComplyToForcesAlongLine, and versions of guarded move etc.
- Repositioning routines: Move the robot to supplied position. The supplied position could be a relative distance, or absolute frame.
- Velocity commands: Move the tool with supplied velocity, again the velocity could be a joint velocity, or tool velocity.

• sensing routines -

- sensor thresholding routines - apply a threshold or a set of thresholds to detect a discontinuit such as contact or breakthrough.
- profile detection routines - monitor the sensor values to check for a predetermined profile. For example a puncture profile contains a monotonic increase in contact forces, a sharp drop, and settling effects. Given nominal maximum forces

• planning routines -

- workspace limits - limit the active workspace of the robot, add minimum and maximum velocity, acceleration limits etc.
- constraints - restrict the motion of the tool based on planning information, modify compliance gains to favor the goal direction etc.

Routines are used as library functions to compose higher level primitives called actions and events. A complete list of implemented routines appears as part of the description of the specification language in appendix A. In the following discussion, we will illustrate the use of this specification language to implement a task graph specification.

Actions are distinctly identifiable portions of a task. An action may involve one or more manipulation routines.

Definition 4.2 (Action) *An explicit specification of a high level sensing, manipulation, planning, for parameters used by the routines, which may then optimized during execution.*

Actions are states of the task's finite state machine. For the manipulation task discussed above, each action consists of a call to a single routine. The *translate* action calls the routine to move base joints *ComplyBaseJoints*, and the *orient* action calls the routine to move upper joints *ComplyArmJoints*. A simple implementation of these actions could be:

```
#action to move base joints
action translate{
        #move the base joints in response to user forces
        ComplyBaseJoints;
} # end of action

#action to move upper joints
action orient{
        #move the RCM/End-effector joints in response to user forces
        ComplyArmJoints;
} # end of action
```

Events form the conditions that must be met before the action is executed and guards that cause the manipulation primitive to yield control.

Definition 4.3 (Event) *A composition of sensing, planning, or safety routines that monitors a detectable change and might effect a change in the manipulation behavior.*

For the above task, three events are needed, two that transfer control between the two actions, and one that allows the user to halt execution. Every action defines its events in an event map, which is monitored while the action is being executed.

Here, these events are *translate2orient* and *orient2translate*. When these routines are simply monitoring the buttons on the foot pedal, they can be defined as:

```
#event map for translate action
events{
        on translate2orient orient;
} #pass control to orient if event detected
#event map for orient action
events{
        on orient2translate translate;
} # back to translate mode
```

This sets up the two state FSM, allowing us to move between *translate* and *orient*. The event to halt execution is monitored in both states, and can be inserted in the global event map, to be monitored at all times.

```
#global event map
events{
        on event2quit done;
} #halt execution
```

The user provides a task skeleton in form of a composition of actions (e.g. for peg in hole, actions for positioning over the hole, inserting, and detecting contact etc.). For cooperative manipulation this task skeleton is a simple and sufficient mechanism for representing task strategies.

Definition 4.4 (Task) *A composition of actions that performs designed to achieve a specific manipulation or sensing goal.*

The skeleton sequence is parsed and used to construct the task graph. A task skeleton contains forward declarations of its actions and events, a task initialization section, and a global event map, and definitions of actions and their event maps, and events. For the above task, this task skeleton looks like:

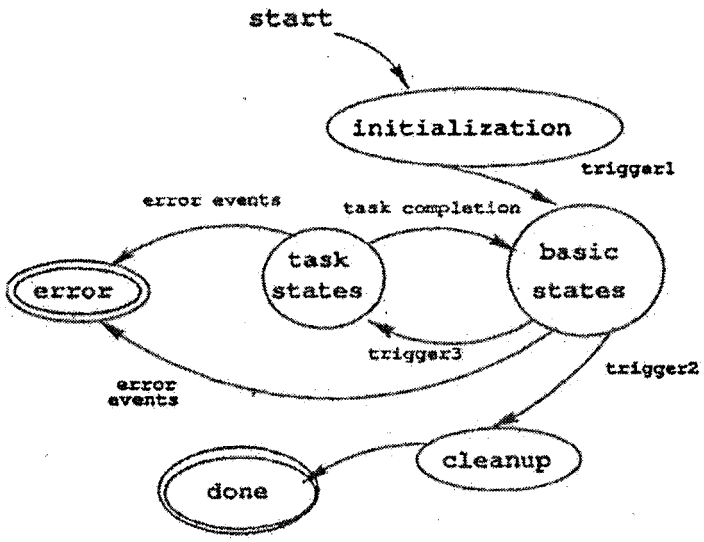

Figure 4.4: System model for executing task graph specifications. The basic functionality of the system allows the user to manipulate the robot. The task states represent the task graph for the current task. The user can choose to execute the task graph or halt execution. During the task graph execution, completion of the task returns control to basic system states.

```
action translate, orient;          # task and event definition
event translate2orient, orient2translate, event2quit;
task simpletask{          # task initialization section
       #initialize the user force sensor:  assume hands-off position
       BiasUserForces;
       #and set default limits on it
       EnableForceLimits;
       #setup the robot to receive motion commands
       Enable;
} #end of task init
```

The event definitions are similar to the defection of the actions. The System

also maintains a basic set of states, and events. These include additional actions for initialization and cleanup, basic steady hand manipulation, data collection, and some safety and error checking. A sketch of the overall system organization appears in figure 4.4.

4.3 Analysis of task graphs

As with discrete event systems in other areas such as models of feature interactions in telecommunication networks, the task graphs can be analyzed for graphs properties. For task graphs under consideration, some analysis of the corresponding state machine as a deterministic finite automaton is straightforward. The analysis below follows the notation used by Lafortune at al [Cassandras *et al.*, 1995]. For other work in control and supervisory control of discrete event systems see [Ramadge & Wonham, 1989, Charbonnier *et al.*, 1999, Chen *et al.*, 2000].

Consider a finite state machine or deterministic finite automaton (DFA) G. It is a six tuple $G = (X, \Sigma, f, \Sigma_g, x_0, X_m)$. where

X is the set of states of G, or actions in our task graph;
Σ is the set of events associated with the transitions in G;
$f : X \times; \Sigma \mapsto X$ is the partial transition function of G,
i.e. $G : f(x, e) = x'$ relates the actions and transitions;
$\Sigma_g : X \mapsto 2^{\Sigma}$ is the feasible event function.
$\Sigma_g(x)$ is the set of transitions possible in state x;
x_0 is the start state;
X_m is the set of final states.

If this DFA models a task graph, we would be interesting in finding out if

- Are there any states in the task graph that do not lead to a successful completion?

- Are there any states in the task graph for which there are no events leading to their execution?
- Is there any state(s) that never yield control to other states?

Since we would like our task graphs to always lead to a finial state, these are cases we would like to avoid. The first case, can be formalized as:

Definition 4.5 (Trim) *A FSM G is said to be trim if all states of G are on a path ending in a terminal (success or error) state.*

The language generated by G is defined by $L(G) = \{s \in \Sigma^*; f(x_0, s) is\ defined\}$, and the language accepted by G is defined by $L_a(G) = \{s \in L(G); f(x_0, s) \in X_m\}$. The specification language for our task graphs restricts states that are unreachable or that don't have a way to termination, the FSM for any parsed task graph (and so any accepted language) is trim. So we are ensured a path to completion.

The other cases involve states that that have no path leading to them, or from these states to a terminal states. These conditions are classically identified as deadlock, or livelock.

Definition 4.6 (Deadlock) *A FSM G is said be in a state of deadlock if there are no transitions from the current state.*

Definition 4.7 (Livelock) *A FSM G enters a state of livelock if there is no path to a terminal state from the current state. This could be due to a cycle in the state-graph that does not contain a terminal state.*

To ensure that the task graphs avoid livelock and deadlock, the task graphs are analyzed to ensure,

- Each state has at least one incoming transition, and a path from the start state.
- Each state, except for the terminal states have at least one path to a terminal state. A task specification with a state that has no incoming transitions, or an incoming transition but no way to transition to a terminal state is not accepted.

- The system automatically adds transitions to error states if none is specified.

The accepted language therefore, requires that each state must have a way of getting executed and the planned execution have at least one way of terminating in success or error.

Task analysis of Basic steady hand control task graph

The task graph for basic steady hand operation in figure 4.3 can be analyzed using the above principles. Consider a deterministic finite automaton (DFA) G_b. It is a six tuple $G_b = (\ X, \Sigma, f, \Sigma_g, x_0, X_m)$.

where $X = \{$ *translate*(t), *orient*(o), *done*(d) $\}$

Σ is the set of events

These events are $\{$*translate2orient*(e_0),*orient2translate*(e_1), *event2quit*$(e_2)\}$.

$f : X \times \Sigma \mapsto X$ is defined completely by $\{f(t, e_0) = d, f(o, e_0) = d,$

$f(t, e_1) = o, f(o, e_2) = t\}$;

$\Sigma_g : X \mapsto 2^{\Sigma}$ contains $\Sigma_g(t) = \{e_0, e_1\}, \Sigma_g(o) = \{e_0, e_2\}, \Sigma_g(d) = \{\}$.

$x_0 = t$ is the start state; and

$X_m = d$ is the final state.

Clearly, there is always a transition to the terminal state on all the paths possible from the start machine. This simple three state machine is non-blocking, and deadlock or livelock is not possible.

4.4 Safety and Error Recovery Issues

The above task graph does not contain any error handling. There is a well defined set of events that addresses basic software and hardware error handling issues that can be easily added to the above task graph.

Common Error Events

The robot is a mechanical device, it may experience faults. Fortunately most of these are well studied. These can be categorized:

- Hardware Errors: this includes motor and encoder failure, amplifier faults and cable failures. These faults usually require manual attention for the hardware and it is common to go to a terminal error state if these occur.
- System Errors: timeouts, and watchdog timers, servo loop errors for the robot etc. These errors occur primarily due to the software malfunction. A good way to deal with these is to reset the error and reinitialize the device causing the error a fixed number of time, but go to a terminal error state if the error keeps reoccurring.
- Task and configuration errors: these are usually dealt with in the task description itself. For example if a task limit is encountered, a common response is to prevent the robot from moving beyond the limit, but if the robot moves away from the limit the event is automatically cancelled.

Dealing with Error Events

Some error events are common to all tasks, these fall in the first two categories of errors and are part of the basic system behavior. These events are monitored in all states, and the transition to the error state are automatically defined as part of the system initialization process.

A task defines its error events in its event map, just as it defines transition events. The error event can be an event defined by the task or a system error (in which case it overrides the system policy), and the error action may be the terminal or non-terminal error action defined by the system, or an error action defined by the task.

4.5 Implementing Task Graphs

A simple specification language was developed for specification of task graphs for augmented tasks. This specification language allows construction and execution of graphs on available or composed routines. A program consists of forward declaration of its actions and events, a task section, and action and event definitions. The body of the task section contains global initializations. The task is followed by a global event map for events that might cause state transition throughout the program. Each action is next given a body. Each action could have its own event map. Events are also given a body.

The specification language supports common basic types, as well as types needed for linear algebra. Most common algebraic and linear algebra operators and mathematical and trigonometric functions are also supported. The robot and several additional sensors are part of the specification language and are tracked as system variables. A detailed description of the specification language and execution environment appears in appendix A.

Chapter 5

Experiments

In this chapter we describe experiments which use task-level augmentation for practical extensions of surgical tasks and to evaluate the effectiveness of using steady hand task-level augmentation in comparison to simpler augmentation methods and unaugmented free hand performance.

5.1 Classification Of Suitable Tasks

Not all tasks are suitable for augmentation. This is due to limitations of computational and mechanical hardware, incomplete understanding of environmental dynamics and a lack of ways to efficiently integrate human expert knowledge in a machine.

However, there are many generic tasks in surgical procedures that may be augmented by a single surgical assistant. Microsurgical tasks such as camera holding, tool guidance and positioning, constrained and guarded motions (force constraints are especially hard to implement without augmentation and very common in practice)are good candidates for simple augmentation. Some of these tasks have well studied analogues in industrial automation. However, there are significant differences between assembly and surgical environments.

A broad classification of simple surgical manipulation tasks is as follows:

- Positioning tasks: primarily require a tool to be positioned/held in a certain

position. For example: holding an endoscope, holding a micro-pipette inside a vessel, moving an endoscope, moving the tool out of the workspace.

- Path following tasks: require a tool to be manipulated along a path either defined from sensory/user input or from a supplied definition. For example: tracking a blood vessel from imaging information, guiding a drill along a constrained path, guide a cut/drilling motion, inserting a needle/micro-pipette/other tools to desired position.

- Compliant motion tasks: move the tool in response to dynamic sensory input. For example, maintaining constant contact forces (e.g. ultrasound imaging), pulling or pushing the tissue with constant forces.

- Tasks unrelated to motion: require sensory input to be interpreted for detection of events. For example: puncture detection, contact detection, computation of distance traveled, control of tool forces.

These tasks can also be classified using the degree of freedom required to implement them (single vs. multiple degree of freedom tasks) or the technology they relate to (manipulation, sensing, computation, planning, registration etc.).

Each selected task can be performed conventionally, cooperatively with the robot and (where applicable) by the robot alone. Cooperatively, several modes of augmentation are possible. The simplest of these modes is the robot acting as a compliant tool. Various complex augmentation schemes using task graphs and user selected parameters can also be used. If these augmentation schemes are considered as a single mode, then each experiment can be performed in the following modes:

- **Unassisted:** only the conventional techniques are used to perform the task.

- **Assisted:** the robot is used to assist the user, but only as a compliant tool holder, with no knowledge of the task.

- **Augmented:** a task graph is used to actively augments the user in performing the task, and where required the robot performs some parts autonomously at the users command.
- **Autonomous:** the robot performs the task without user input.

In all experiments, the users are allowed unlimited training time until they are comfortable with the experimental setup. Each user then performs the task several times. For both robotic and free hand experiments , a blind data collection system is used to evaluate the performance. However, in some cases visual observation by the user or an observer becomes necessary.

5.2 Performance Measures

The goal of these experiments is to analyze human performance for a range of common motions that form parts of surgical tasks. A measure of completion without error forms the main criterion for evaluating the performance. This could also be measured in terms of the number of tasks completed in the given time period.

A comparison of the number of errors to the number of attempts gives a measure of accuracy. Comparing the successful attempts to the number of total attempts gives a measure of effectiveness of the augmentation and the ease of use. Comparative performance for longer periods of time measures stress and user fatigue.

From a surgical perspective, other performance criteria should be measured. Tool positioning tasks have certain accuracy requirements. Performance in both absolute tool positioning and constrained tool motion are good measures of the performance of these tasks. These can be measured either in terms of the average error in positioning, or the largest error over a period of time for a positioning tasks. Similar measures of performance can be applied to path following and constrained motion tasks, along with a measure of difficulty of the constraints used.

Tasks that do not relate to motion can be measured in time. This includes comparison of the average time of detection. Minimum and maximum times of detection

gives bounds for such a task.

Many surgical tasks combine positioning, guidance, and parts that do not relate to motion. In these tasks, each portion can be compared individually along with measures of success and times of completion.

Subjective evaluations by the user provide indication on the acceptability of the system. Most performance studies include subjective evaluations for ease of use, comparison to clinical settings, amount of training required etc. In our experiments, each user also evaluates the experimental setup subjectively on ease of operation, seating comfort, and ease of viewing the target.

Individual experiments may allow other performance criteria.

5.3 Experimental Tasks

We have selected a range of common manipulation tasks below. The first example is a peg in hole task. It is used often, e.g. [Salcudean *et al.*, 1997], [Das *et al.*, 1999] to compare positioning performance. We have also used this task [Kumar *et al.*, 1999] in past experiments. It is also a good vehicle for explaining the experimental concepts. More complex tasks involving constrained positioning, virtual fixtures, and force scaling are examined in later sections to illustrate the board scope of application.

5.3.1 Peg in Hole tasks

In this experiment, we compared the accuracy and reliability of performing a highly precise positioning task. The goal was placement of a microsurgical probe into holes of various sizes under magnified vision. In surgery this task is similar to performing a precise biopsy or positioning a needle for suturing a small vessel.

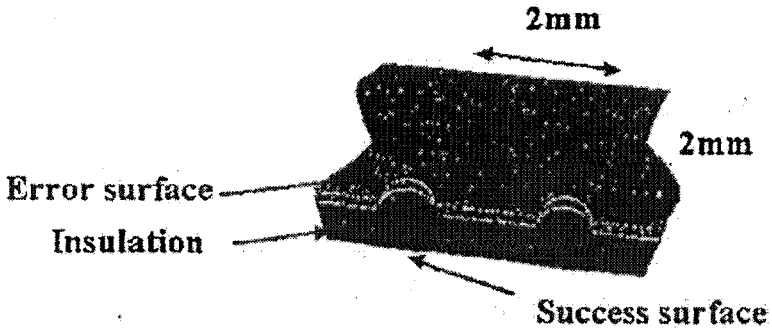

Figure 5.1: The data surface for peg in hole experiments. The top error surface and the bottom success surface are separated by 50μm insulation, with the total depth of the hole being about 150 μm. A grid of holes with 2mm spacing is interleaved with a grid of 8mm spacing for 250μm, 200μm, 150μm hole sizes.

Experimental apparatus

A 10-0 microsurgical suture needle held with the help of a needle holder was used as the probe and a datum surface provided the holes. The operating microscope used as a visual aid provided with up to 40 fold magnification. The datum surface consisted of a sandwich of two metallic sheets separated by an insulating surface. This sandwich contained grid patterns of holes of the same size placed 8 and 2mm apart in interleaved fashion. These were duplicated for the three different sizes of the holes (250μm, 200μm, 150μm). The thickness of the upper (error plate) was 100μm, and the insulating surface was 50μm thick. This made the holes 150μm deep (figure 5.1).

The lower surface was the success surface, and the upper the error surface. Both surfaces were connected to I/O lines as was the microsurgical needle. Any closure of circuit generated a success or error event which was accounted automatically.

Augmentation Strategies

The peg in hole task is perhaps the most studied task in robotics. Several detailed analyses are available in the literature that guarantee successful insertion once the

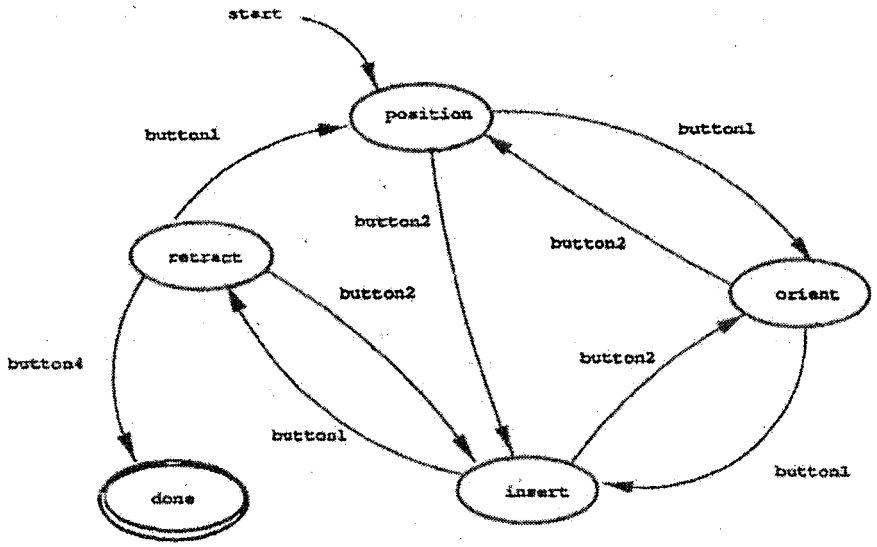

Figure 5.2: A Peg in hole task graph. The user uses the base stages of the robot to roughly position the robot, then uses the rotational degrees of freedom to orient the probe before inserting in the hole. In this version of the task, the user signals the transition by using the foot pedal.

peg is partly in the hole. Most of these analyses deal with uncertainty in the position and size of the hole. Lozano-Pérez et al [Lozano-Pérez *et al.*, 1984] suggest a variety of strategies. These include chamfers, tilting the peg, a surface search by sliding the peg on the surface, or a biased search with a bias into the initial position.

These strategies assume contact with the surface is allowed at points other than the target hole. They also assume that the hole is located at an estimated or approximately known initial position. These assumptions do not hold for most surgical manipulation tasks. The user decides where he wants the tool to go during execution, and it is an error to make contact with the tissue at any other place. Further minimally invasive tasks constrain the tool to move while keeping the insertion point on the skin surface or port fixed.

Analysis of the peg in hole task suggests a simple heuristic strategy for augmentation. Let us first ignore the minimally invasive constraints. The users move the

Hole size	Experiment Mode		
	Unassisted	Assisted	Autonomous
250 μ m	48.8%	77.8%	98.4%
200 μ m	46.3%	77.6%	97.7%
150 μ m	43.0%	79.0%	96.5%

Table 5.1: Augmented modes improve performance significantly for a precise positioning task.

probe close to the hole, once near they carefully align the probe above the hole before inserting it in the hole. This suggests a three action task. The first action (*position*) has little possibility of error, and users comfortably position the probe over the hole. The second action (*align*) requires they move at much slower velocities to align the probe with the hole and avoid errors. The third action (*insert*) is almost pure insertion with some adjustment to the alignment of the probe. This simple task graph appears in the figure 5.2.

The other strategies for a peg in hole task can be used if more information, e.g. the position or spacing of holes (desired distance between suture holes), is known. Further, guiding the user towards the hole would also be possible. Strategies relying on contact sensing are not appropriate here. A simple contact search strategy could be moving the probe towards the hole till it is in contact with the surface and then slide it in the hole. However, for a microsurgical task since making a contact with tissue in any place other than the target would be an error.

Minimally invasive constraints would require constraints on the RCM point to remain fixed at all points when the tool tip is below the RCM point (i.e. inside the body). This modified task graph is discussed along with experiments for vein cannulation in section 5.3.2.

Hole size	Experiment Mode		
	Unassisted	Assisted	Autonomous
250 μ m	65.0	29.6	5.0
200 μ m	76.8	27.0	7.0
150 μ m	66.8	25.0	11.0

Table 5.2: Average number of erroneous attempt for a peg in hole task over 10 minutes for six users. The assisted mode reduces the number of errors significantly, Autonomously, the robot makes very few errors. These errors may be due to errors in registration and fabrication of the data surface.

Results

From the current data, the number of errors per successful try improve significantly with the robot. The total time for the task also decreases when the robot is used.

Data from six users appears in table 5.1. It compares the success rates (percent of success in total attempts) for the experiment. The average *unassisted* success percentages were 48.8, 46.3, and 43 percent for the three hole sizes (single factor ANOVA, $p<0.0001$). When *assisted* by the robot the success rate improved to 77.8, 77.6, and 79.0 percent successes. In this case, the size of holes did not significantly influence the performance. [1]

The number of attempts, and success percentage remained approximately constant for the augmented modes. Because of superior accuracy of the robot, the decreasing size of holes did not pose a significant problem.

The robot was registered to the data surface for the autonomous mode, and outperformed cooperative modes. Better performance however, is not unexpected. A precise robot is likely to outperform any cooperative mode, if the task is completely specified in the robot space.

[1]The same experiment with the LARS robot, a laparoscopic assistant, with positioning accuracy of about 0.1mm resulted in 56, 50.8, and 46 percent successes (Single factor ANOVA, $p<0.02$) and reduced the errors significantly (paired t-test, $p<0.01$). Size of holes did not significantly influence the result.

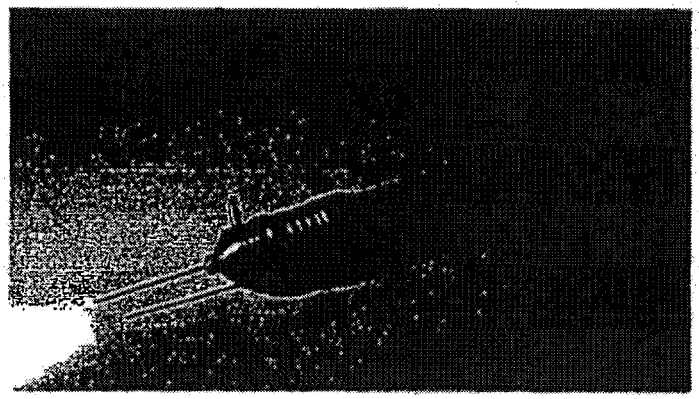

Figure 5.3: The GRIN lens endoscope (Insight IE 3000).

5.3.2 Constrained Motion Tasks

In several surgical tasks, especially those involving a port access, motion constraints are automatically imposed by the anatomy. Laparascopy, and endoscopy are two easy examples. Laparoscope's and endoscopes only visualize a small part of the area of interest at any time, and need to be moved to view the entire region of interest. This task becomes harder when good images can only be obtained by locating the imaging device at a small distance from the anatomy.

Application: Retinal Image Mosaicing

Diagnostic retinal imaging presents a very good application of our system. Retinal fundus images are a common mode of diagnostic imaging for the eye. Endoscopic imaging using small probes (e.g. GRIN lens endoscope, Insight Instruments Inc. , figure 5.3) can be used to provide higher resolution imaging of small regions of interest. However, these probes only image a very small region (few mm^2) and need to be held at similarly small distance from the retina. Creating a compound image of the anatomy without robotic assistance is nearly impossible. Using robotic augmentation, this is a constrained guidance task. Further, irregular regions of interest can be selected.

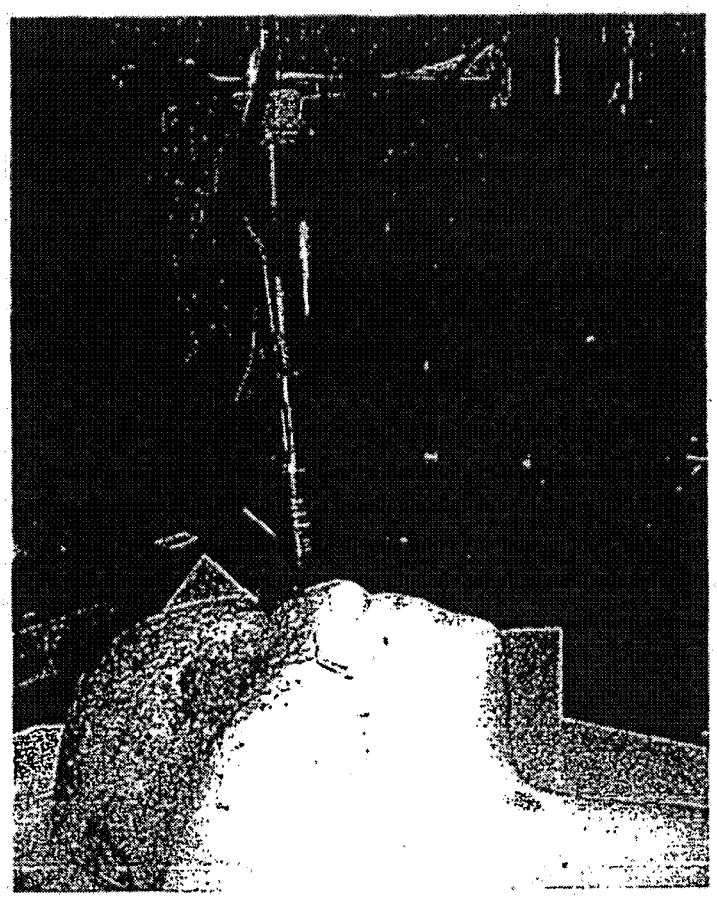

Figure 5.4: The GRIN lens endoscope (Insight IE 3000) experimental Setup for mosaicing experiments. The force sensor and ergonomic handle used for manipulating the robot are mounted at an offset and appear at the top.

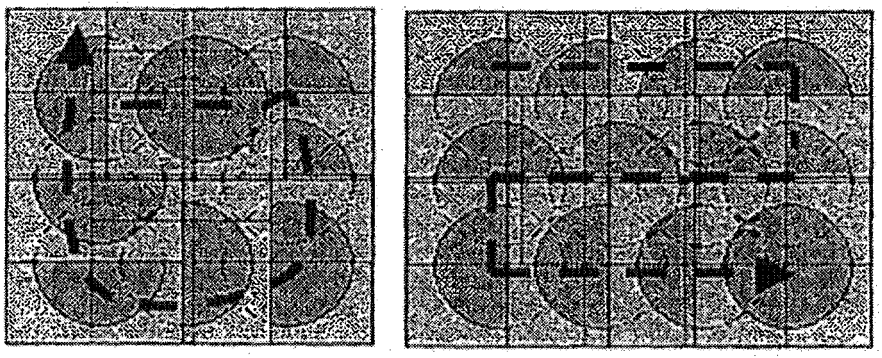

Figure 5.5: Augmentation strategies for creating retinal mosaics. A spiral movement centers the collected images about the region of interest, whereas a grid pattern may be more suitable if a long and thin region, such as a single vessel is to be imaged. The task graph allows the user to interactively move the robot to the next image position.

Experimental Apparatus

For this experiment, the GRIN lens endoscope (figure 5.3 Insight IE 3000, Insight Instruments Inc.) was mounted on the insertion stage of the steady hand robot. The ergonomic handle used for sensing the manipulation forces was mounted at an offset using an adapter. This configuration can also be used to mount endoscopes of a larger size for neurosurgical tasks.

A porcine eye was positioned naturally in a phantom head, and a port was created in the eye for positioning the endoscope such that the RCM point was at the port. In some cases, the eye's lens was removed to expose the retina and provide better access. The experimental setup appears in figure 5.4.

Augmentation Strategies

The goal of the task is to completely image the selected portion of the retina in an efficient way. If the region of interest is a block region (figure 5.5), a simple strategy could be a spiral movement starting with the center of the region of interest. On the other hand, if we consider imaging along a retinal vessel only, then moving in a grid fashion to capture long strips is a possibility. In either case, the user selects

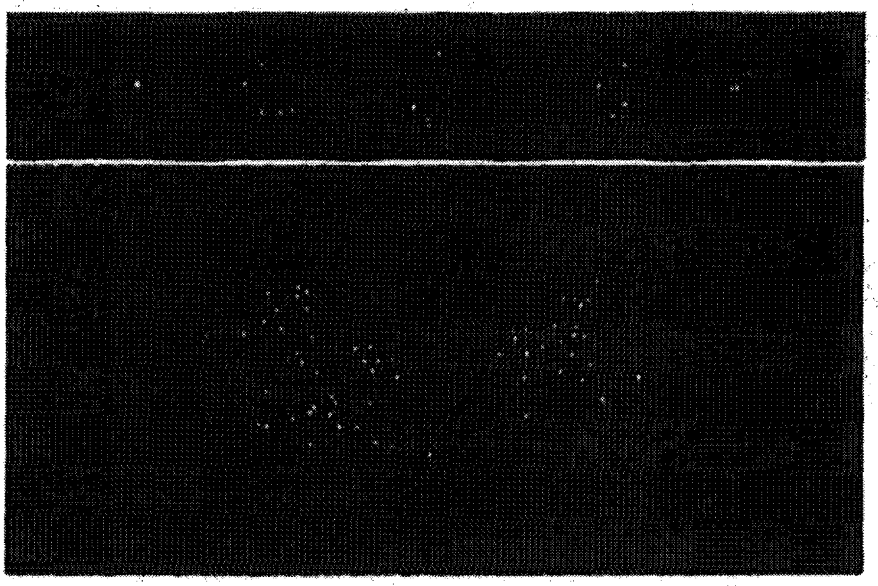

Figure 5.6: A collection of 5 GRIN endoscope porcine retina images (top) taken in a series and the composite image(bottom).

the amount of motion between the consecutive images to ensure properly overlapping images.

A task graph for the mosaicing task appears in figure 5.7. The scope is first positioned in the eye and one of the above augmentation strategies is activated. The user then moves the robot to the next step, images the region and repeats the process until the entire region is imaged. The scope is then retracted to finish the task.

The user navigates the spiral or grid, using the ergonomic handle by applying forces on it. The action *increment* implements the following algorithm;

```
if(force-applied)
    while(position < next_step)ComplyToUserForce;
    next_step = next_step + increment;
```

The increment vector is set to the width of the image at either end of the grid for one image, and changes sign at either end of the grid. For a spiral, it alternates between the width and height for an equal incrementing number of images.

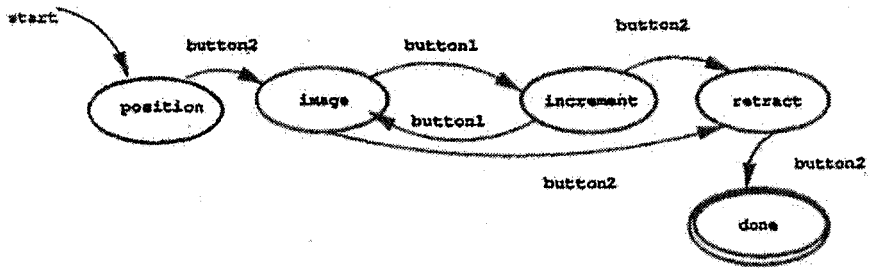

Figure 5.7: Task graph for creating retinal mosaics. The action *increment* implements one of the augmentation strategies discussed above. The user begins by positioning the endoscope in the region of interest and then moves the robot and signals for image capture until the region of interest is imaged.

Results

The images and the robot position at which they are taken are then used to construct the composite image. Because of the accuracy of the robot, even a simple reconstruction procedure provides good high resolution montages. An example of a line of 5 images and the corresponding composite image appear in figure 5.6. A larger grid of images and the composite image appear in figure 5.8. In these case, the eye lens was removed and translation stages were used for motion.

Application: Retinal Vessel tracking

If an endoscopic instrument is used to locate a blood clot in retinal vessel or otherwise diagnose the state of blood vessels on the retina, the user needs to move along the blood vessel. The blood vessel can be modeled as a simple curve and this is also a constrained motion task. Retinal vessels can be tracked in the images produced by the endoscope to restrict the motion along the vessel.

Experimental Apparatus

In an initial experiment for tracking simple curves and lines, a neuroendoscope was used with the steady hand robot, and the ergonomic handle was attached at an

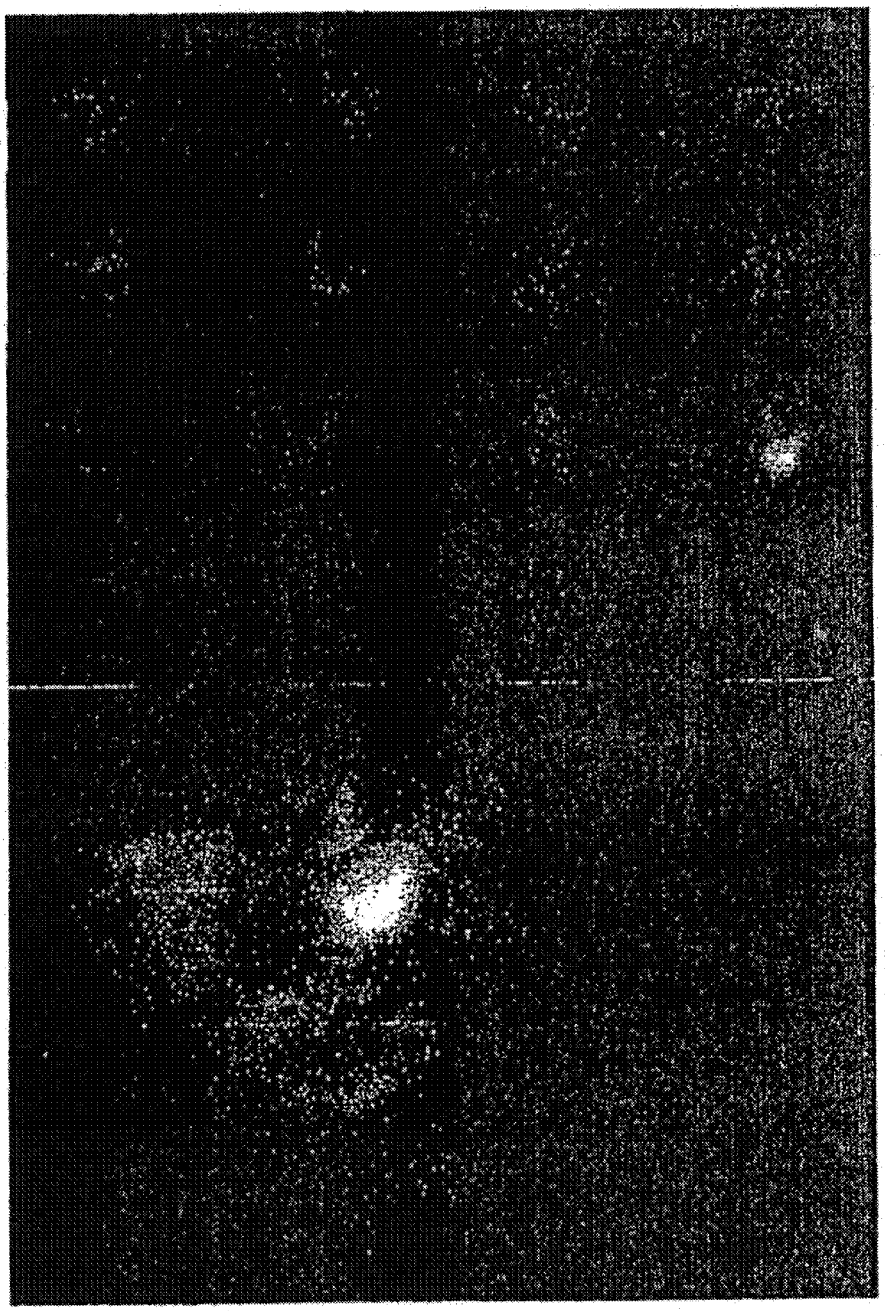

Figure 5.8: A larger collection of porcine retina images, and the corresponding composite image. These images were taken using a gird of overlapping images.

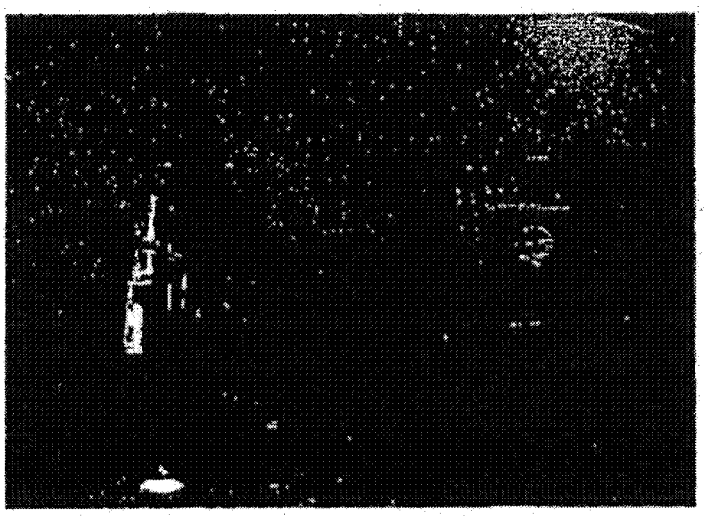

Figure 5.9: A simple experiment setup for tracking lines and curves. A neuroendoscope is used with the steady hand robot to capture images, and the force handle is mounted at an offset.

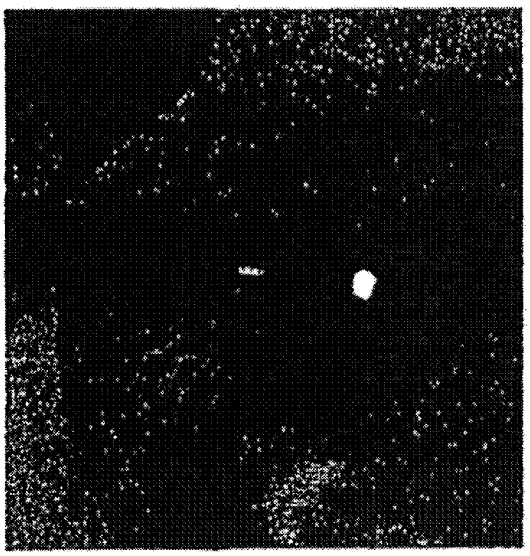

Figure 5.10: The GRIN lens endoscope used for vessel tracking. The porcine eye is mounted in a foam face phantom with its lens is removed for better access. The robot is then guided to position the endoscope appropriately.

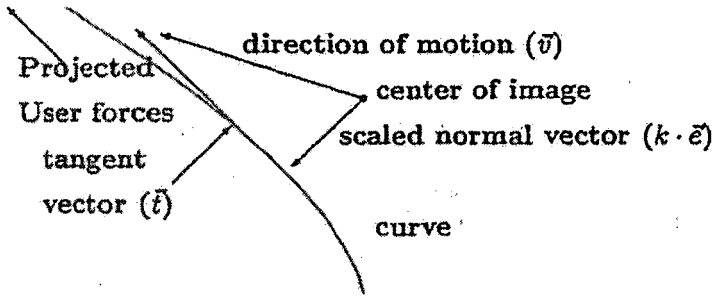

Figure 5.11: A projection of the user forces on the normal to the curve is added to the scaled normal distance of the curve to guide the user along the curve. The robot moves to center the curve in the image and constrains the user to move only along the vessel.

offset for capturing user forces (figure 5.9).

The second experimental setup was the same as the mosaicing experiment. The images from the GRIN lens are used to track the position and orientation of the blood vessel, the robot then restricts the motion to position the vessel in the center of the screen. The user selects a blood vessel and then starts tracking by pressing the foot pedal. The user is then restricted to move only along the blood vessel.

Augmentation Strategies

Consider the vessel to be modeled as a straight line. If the user is constrained to move only up and down the line. The direction of motion (v) is the sum of the normalized tangent vector ($\vec{t}$) to the line, and scaled normal distance ($k \cdot \vec{e}$) of the line to the center of image (figure 5.11). By moving in this direction ($\vec{v} = \vec{t} + k \cdot \vec{e}$) the user centers the line in the image, and then moves along it.

If the update rates from the image processing are slow, instability is possible. However for a range of velocities, and k stable tracking can be achieved.

The task then can be broken down into the following actions. The user positions

Robot Joints Used	Experiment Mode		
	Assisted	Augmented	Autonomous
Translation	1.1mm, 1.6mm	0.95mm,1.75mm	0.03mm,0.22mm
Orientation	0.61mm,1.04mm	0.37, 1.12mm	0.26mm, 0.47mm

Table 5.3: Average, maximum errors for tracking a line. Automatic tracking of a line results in very small errors. Augmentation with the scheme in figure 5.11 also results in smaller errors than a compliant motion mode, but does not compare with automatic tracking if the user is still allowed to deviate from the curve. Augmented performance compares with automated execution if the user is not allowed to deviate from the curve.

the endoscope to a port in the porcine eye (*position*), and then orients it to be able to insert it through the port (*orient*). He then inserts the endoscope to the appropriate distance above retina and moves it so that the target vessel is in view (*adjust*). Once the vessel is in view, the direction of motion and down the vessel is preferred (*constraint*) using the technique in figure 5.11. This task graph is similar to the task graph to the task graph for creating mosaics in figure 5.7.

Results

For the experiment in figure ??, lines and curves drawn on paper were tracked. Since there is no minimally invasive constraint, both translation and rotation joints can be used. Starting with the line centered in the display, results from tracking a 1mm wide line by looking at the video display and moving along it appear in table 5.3. Two modes of the robot were used separately, the base translation joints (*translation*) and the upper rotation and end-effector joints (*rotation*). Again, the robot outperforms other modes automatically. Augmented mode, where the robot assists the user by applying directional gains in figure 5.11, but with a large k still allowing the users to deviate from the line has smaller average errors than a compliant motion *assisted* mode.

Augmented strategies achieve acceptable results for lines and simple curves. The

robot *autonomously* is able to perform better tracking than other modes. It is more difficult to for users to traverse these curves using simple compliant motion and larger maximum errors are noticed. Augmented versions reduces the error.

Virtual fixtures are an attractive means of guiding a user using planning information, and information extracted from the sensors. Bettini et al [Bettini *et al.*, 2001] have further investigated using virtual fixtures with the steady hand approach. They track lines and simple known curves drawn on a paper, with a camera mounted in a similar configuration. Their work extends this work and provide other approaches for constrained direct manipulation.

5.3.3 Dexterous manipulation tasks

The above examples form portions of many complex tasks. A combination of the above strategies yields a strategy for performing a constrained precise positioning task such as puncturing a blood vessel in the eye (retinal vein cannulation).

Retinal vein cannulation

A retinal vein occlusion [Kohner *et al.*, (1983] is essentially a blockage of the portion of the circulation that drains the retina of blood. With blockage of any vein, there is back-up pressure in the capillaries, which leads to hemorrhages and also to leakage of fluid and other constituents of blood. The diagnosis of a retinal branch vein occlusion poses little difficulty to an ophthalmologist who will detect dilated blood vessels, hemorrhages, and swelling (edema). It appears that the more complete the blockage, the more intense the hemorrhages and the edema.

There is no known medical treatment for retinal vein occlusion. Anti-coagulants such as heparin, coumadin and aspirin have not been shown to be of value in preventing branch vein occlusion or managing its complications. Because anti-coagulants may be associated with systemic complications, they are prescribed only in specific clinical circumstances, for example for patients with known clotting abnormalities.

Local application of anti-coagulants has been investigated as a treatment. But

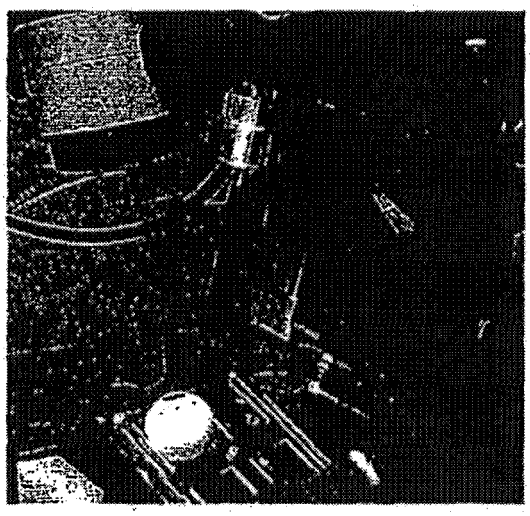

Figure 5.12: A constrained peg-in-hole task simulating vein cannulation in the eye. A ball is sliced to simulate the constraints in eye surgery, and a data surface is used to measure human ability to place a tool accurately in the simulated eye.

due to the small size of the retinal vessels, it has not been feasible to deliver anti-coagulants to the blocked vessels. It may be possible to use the steady hand robot to make local delivery of anti-coagulants possible.

Experimental Setup

A setup for comparative performance experiments appears in figure 5.12. To construct the target a ball was sliced and ports constructed to reflect distances similar to the eye. This ball was attached to a data surface containing 100 micrometer holes separated by 2mm. An ergonomic tool handle was mounted with a 1 mm shaft and 50 micrometer tip wire for the tool. The goal of the experiment is to touch the bottom of the hole inside the "eye" without touching the sides. Automatic electrical contact sensing is employed to detect contact between the bottom of the holes *(success)* or the sides of the hole or elsewhere on the surface of the data surface *(error)*.

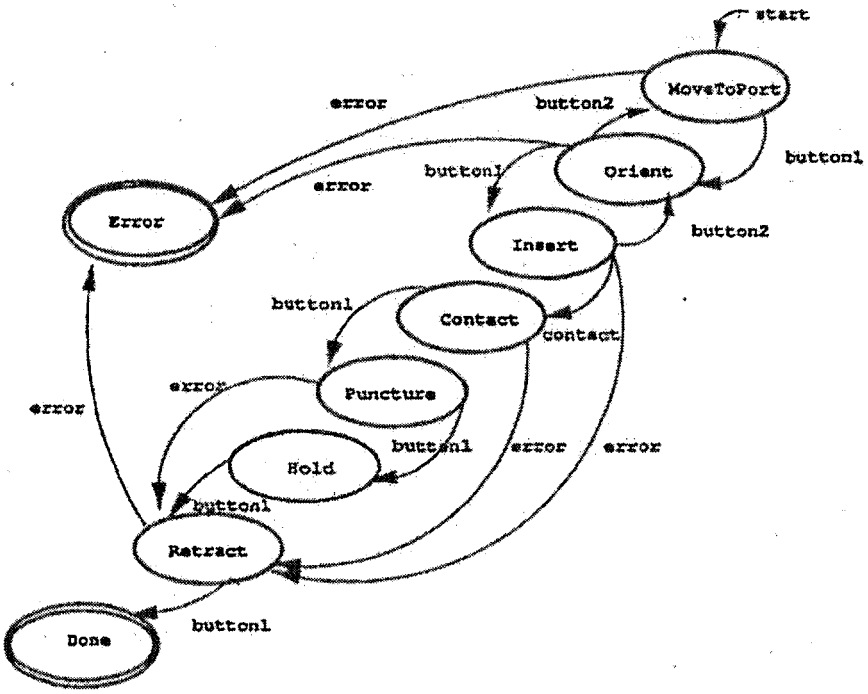

Figure 5.13: Task graph for retinal vein cannulation.

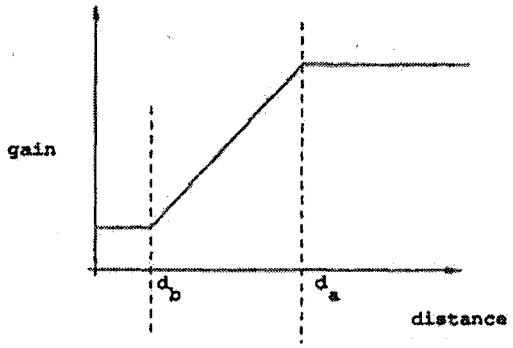

Figure 5.14: Gain profile for augmented constrained peg-in-hole experiments. If the distance to target is known then the gains for force control can be modified so as to assist the user in moving to the target. The user finds the robot stiffer as he approaches the target. Since the task requires higher accuracy at the target, this strategy is more suitable than one that allows higher gains at the target, and lower as he moves away.

Augmentation Strategies

This is a constrained tool placement task that presents many of the fine manipulation difficulties encountered in eye surgery. It is also a modified (or minimally invasive) version of the peg-in-hole task. This task is also similar to several common invasive tasks. The task of placing a tool through a port (small incision or hole) at a surgical site is one of the most common tasks in laparoscopy. The task is composed of the following steps,

- positioning the tool at the port;
- re-orienting it such that it can be inserted;
- insertion of the tool;
- adjusting the orientation of the tool towards the placement site while viewing through the microscope;
- approaching the site;

- achieving contact;
- puncture;
- hold to deliver therapy, and
- retract the tool outside the eye.

This task has both coarse external manipulation (positioning and orientation leading to the port) and fine manipulation inside the organ. Further if the port is taught to the robot, a tracking scheme can be used to assist the user to guide the robot to the port. The gains for each action can be modified as a function of distance to the target for that action (figure 5.14). The target for coarse manipulation actions is the center of port on the ball, and for fine manipulation a selected hole. The port and target locations are taught to the robot by hand guiding it to both locations.

If the tool were instead used through the working channel of an endoscope, then the vessel could be identified, and augmentation can be used to help the user along the vessel.

A task graph for the task appears in figure 5.13.

Results

From the current data (figure 5.17), the success rates for this experiment (number of errors per successful try) improve significantly with the robot, and augmentation adds to the improvement. The total time for the task also decreases when the robot is used. Data from three users appears as a graph below. User 1 had the maximum training time and experience with the system, and user 2 and 3 are familiar with the system. Training time clearly affects the performance, and further experimentation is needed for analysis and evaluation. These experiments are scheduled.

For feasibility studies, a porcine eye is mounted naturally in a foam face phantom and a MEMS needle (XACTIX Inc.)[2] is mounted on a 2mm shaft to be used as the

[2]XACTIX Inc. generously donated a set of microneedles to prototype this experiment, the author thanks them for the help.

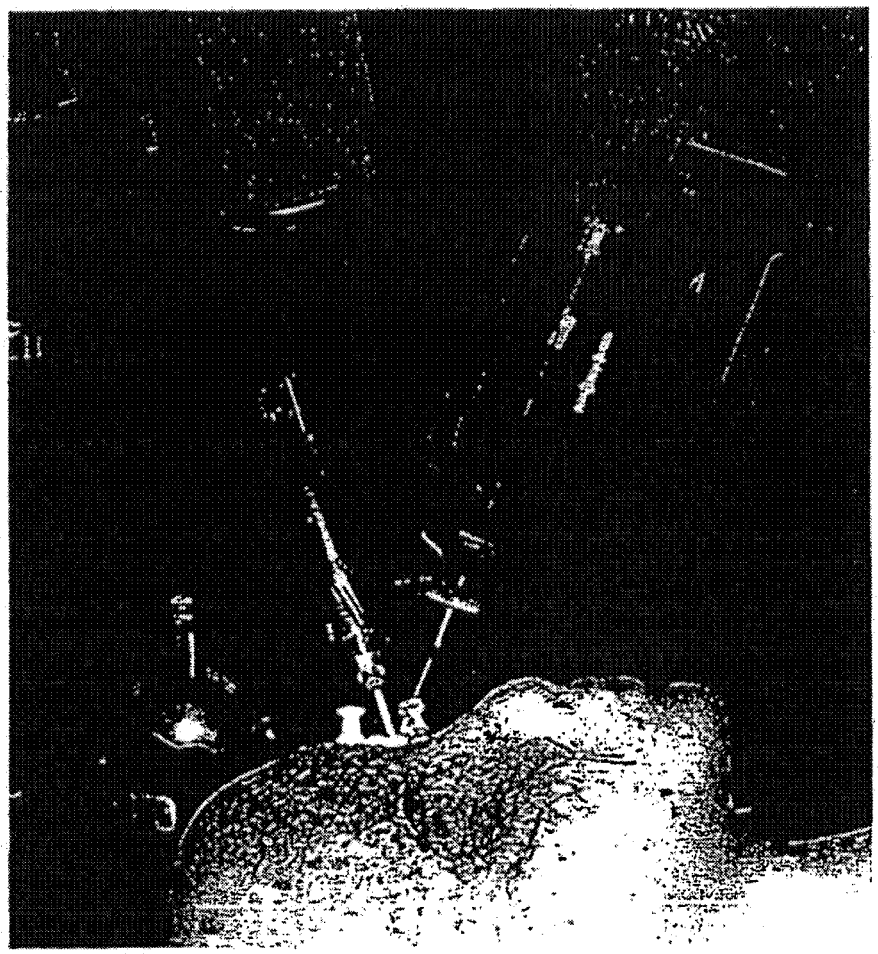

Figure 5.15: Feasibility experiment for retinal vein cannulation. A porcine eye is setup in a foam face. A microneedle is mounted on the ergonomic handle. The GRIN endoscope is used for visual verification. The microscope used by the user is visible at the top of the image.

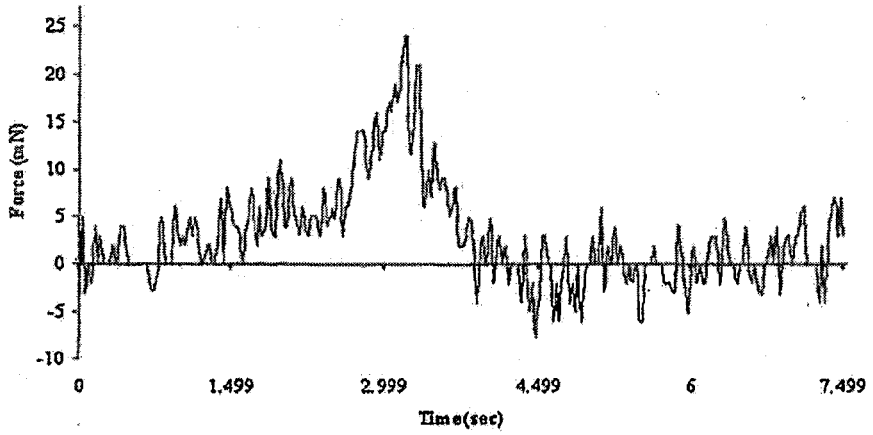

Figure 5.16: A sharp microsurgical probe puncturing a retinal vessel experiences small axial forces. A distinctive peak was visible in our experiments but frictional forces on the needle were negligible.

probe (figure 5.15). The MEMS needle approximates, a micro-pipette used clinically to deliver anti-coagulants. Although the needle can not deliver therapy, it still allows evaluation of the augmentation strategies. The lens of the porcine eye is remove to improve access to the retina. The GRIN endoscope is positioned to image the target blood vessel. The goal of this experiment is to position the micro-needle in a retinal blood vessel.

Vessel Puncture

The forces experienced by a needle as it punctures a hollow plastic tube approximate forces similar to the soft tissue punctures [Bzostek *et al.*, 2000] because of the relatively high tensile strength of the plastic material. Porcine eye vessels (figure 5.18) on the other hand only show a single peak corresponding to the puncture and no hold or friction forces. This is likely due to the thin vessel wall, thin and sharp microneedle, and the hollow region beneath the vessel wall.

The single peak, in 10s mN is sufficient to detect puncture automatically by tracking the monotonous increase in force after the contact and waiting for the sharp drop

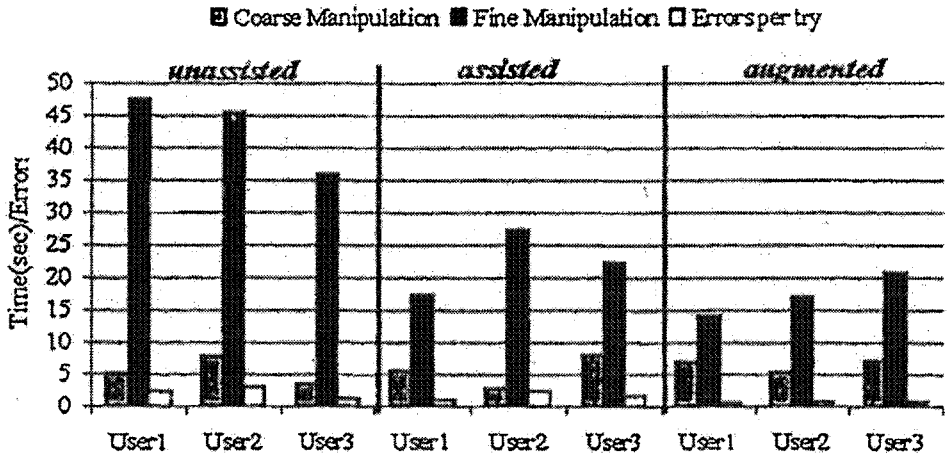

Figure 5.17: Initial results for constrained peg-in-hole task. The number of errors as well as total time decreases with the robot.

in the contact forces. This simple strategy works well in practice for automatic punctures.

The task graph in figure 5.13 was also used on a small hollow plastic tubing, essentially to verify contact, and puncture routines. These routines perform satisfactorily, and figure 5.18(left) shows the micro-needle positioned in the plastic tube. However, in vitro experiments with porcine retina proved harder. A typical targeted retinal vessel is 100s μm, and the lack of blood flow in the porcine eyes reduces the inner diameter to very small values. Further, it also reduces any resistance the passing blood would have provided to the needle. As a result, this first experiment only saw intermittent success in puncturing the outer vessel wall only, and not puncturing the retina. One of the reasons failure was the near vertical approach in these experiments. Surgeons use close to tangential approaches when inserting probes in blood vessels. Similarly, when the approach path of the robot is near tangential or at a small angle, higher success rates are observed. However, the lack of blood flow and partial collapse of the blood vessels remains a problem. These problems can be resolved if an animal model is used instead.

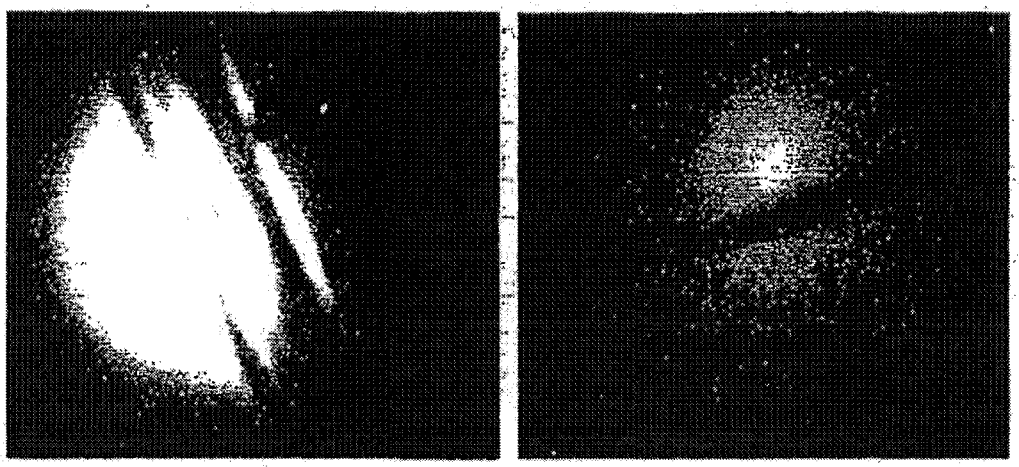

Figure 5.18: Robotic puncture of a plastic tube and retinal vessels. The plastic tube applies sufficient forces on the needle to approximate a force profile very similar to soft tissue organ punctures, where as retinal vessels exhibit only a single peak (figure 5.16).

Several additions can be made to the initial task graph, to improve it. The first improvement is an augmentation of the task graph to enable tracking blood vessels. The second is a change in the experimental setup that improves the success rate.

If the microneedle to be used was inserted through the visual channel of a visualization device (such as the GRIN endoscope), the additional augmentation can be added. Since blood vessels are simple curves, we can use tracking heuristics similar to section 5.3.2 to allow the user to move along a blood vessel once he has identified it. This augmentation appears in figure 5.19. The user is allowed to move between a tracking state, and exiting orientation and insertion states. In the tracking state, the user can both orient, and insert while the robot applies constraints to restrict the motion to only along the vessel. In either state, if the contact is detected motion is stopped to wait for user signal to puncture. This addition allows the user to select a vessel, and move along it. The user is still free to exit the tracking states, and go back to the previous graph if he wishes to select another blood vessel, or is unsatisfied with the tracking performance.

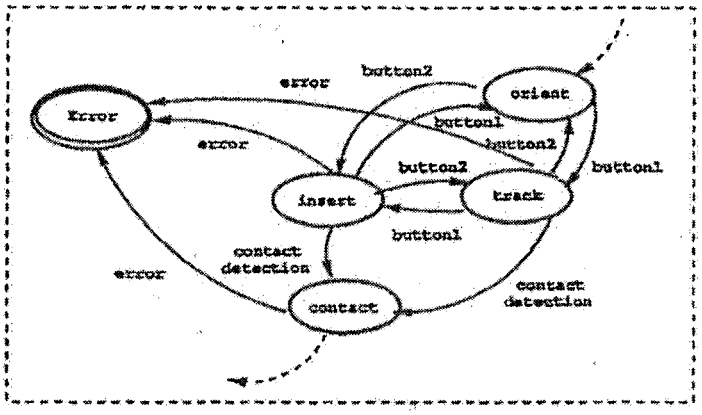

Figure 5.19: Adding a tracking state to the cannulation task graph. The remaining graph is same as figure 5.13. The user can activate a tracking strategy by going to the *track* state, or choose to perform the procedure as before.

5.4 More tasks

In ear surgery, the task of performing a fenestration in the stapes bone to place a piston like implant [Berkelman *et al.*, 2001] is an example. In spine surgery, drilling a hole in a lumbar vertebrae for pedicle screw placement is another example. Since the vertebrae can be registered accurately unlike the soft tissue, the location and the orientation of holes can be planned as well. There are scores of other single instrument surgical tasks that can be modeled similarly.

Chapter 6

Conclusions and Discussion

This dissertation explored task-level augmentation using a cooperative manipulation paradigm. An approach for augmenting cooperative tasks by using a graph based representation of the task was presented. The task graphs constructed using this approach were analyzed as discrete event systems. We used common surgical tasks to illustrate the large set of applications that can benefit from such simple augmentation. The tasks chosen themselves also extend what is currently possible. This work had the following goals:

1. *Create robust primitives for common motions and ways to compose them.*

   A large set of robust primitives for cooperative manipulation were created. These included primitives for compliant motion and force scaling. Sensory primitives for simple actions (e.g. contact, puncture) were also developed. A wide array of sensors including foot pedals, buttons, force sensors, vision sensors, and analog and digital inputs was used for sensory primitives. The information from the sensors could be resolved in relation with the robot. It could also be filtered, biased and scaled automatically. This sensor information was also used to construct virtual fixtures, safety primitives, as well as manipulation primitives. This library of primitives was then used to create a specification language. Sensors were integrated as tracked objects in the specification language. This simple specification language allowed composition of these primitives to create

executable representation of simple cooperative manipulation tasks.

2. *Implement augmentation strategies on these primitives.*

   Several common manipulation tasks were studied and augmentation strategies for these tasks were identified. These strategies were used to construct task graphs, and the equivalent programs were implemented in the specification language.

3. *Integrate explicit representations of non-sensory information.*

   Manipulation requirements of each task were identified and used in the implementation of the task graphs for the task. Virtual fixtures for guidance, and workspace limits were also explored.

4. *Evaluate human performance with and without augmentation for implemented tasks.*

   For some of the selected tasks, experiments were performed to evaluate human performance with varying degree of augmentation. Four modes for the experiments: *unassisted, assisted, augmented* , and *automated* were identified. The tasks were executed in these modes to compare human performance. The experiments show improved performance in both simple and dexterous tasks with augmentation.

6.1 Augmented Steady Hand Manipulation

Steady hand manipulation approach provides a safe, intuitive means of augmenting the manipulation capabilities of a surgeon. It is safe, as the surgeon has direct control of the manipulator, and so of their accustomed surgical tools. It is intuitive, as the surgeon not only directly manipulates those tools, but also receives direct force-feedback from the manipulator, thus restoring the tactile sense of manipulation much as one would during a larger-scale surgical intervention. Other benefits gained from the steady hand approach include its simplicity, straightforward integration into existing application environment and greater immediacy for the human operator.

6.2 Task Modeling and Analysis

There are a large set of simple surgical tasks, for which task models can be constructed easily. We have constructed task models for tasks in eye surgery. These tasks are simple extensions of currently used surgical tasks and practice. Similar tasks exist in other surgical areas. Pedicle screw placement in spine surgery, and drilling of the stapes bone in ENT surgery can be cited as examples.

Analysis of these task graphs using techniques in discrete event system is useful in determining unsafe conditions like deadlock and livelock. Further, work on supervisory control of these systems can be applied for control policies on these graphs as well. We discuss these extensions in more details in the next section.

6.3 Specification language and execution environment

We implemented a simple specification language that allowed us to specify and execute these task graphs. Basic linear algebra and other requirements of a high level robot programming language such as affixment were implemented. Other techniques specific to task graph analysis were also implemented.

Our approach provides several benefits apart from the inherent benefits of cooperative manipulation. It affords the benefits of high-level programming, including easy definition of a task strategy, improved and customized user interfaces, use of same strategy for different tasks, etc.

6.4 Experiments

A large mosaic created using a minimally invasive procedure appears in figure 6.1. The endoscope was inserted into a porcine eye with the RCM point at the port constructed in the eye, very similar to the current manual usage of the endoscope.

Figure 6.1: A collection GRIN endoscope porcine retina composite image. The endoscope was inserted through a port with the remote center of motion constraint and the rotational and end-effector stages were used to manipulate the robot.

The rotation and insertion joints alone were then used to construct the mosaic. A procedure similar to this could be very useful in diagnostic imaging.

Our *in-vitro* experiments are very close to current surgical practice. The tools used in these experiments, are either used clinically such as the endoscope and the foot pedal, or simulate the surgical instruments closely. Therefore, extension of these experiments to *in-vivo* environment, and eventually clinical use should be easy.

Chapter 7

Extensions And Scope For Work

In this chapter we describe limitations and possible extensions of this work. Additions and modifications to this work could make it more practical and useful. Areas for these extensions include the modeling and analysis of surgical tasks, adding automated task state segmentation, and more experimental tasks.

Task Modeling and Analysis

For any task graph, the set of events Σ can be partitioned into two parts: Σ_c the set of controllable events and Σ_u, the set of uncontrollable events. The uncontrollable events represents failure and other unpredictable/unmodeled events that a controller may not be able to prevent. We can then consider supervisory control policies, that can activate and deactivate some of the controllable events.

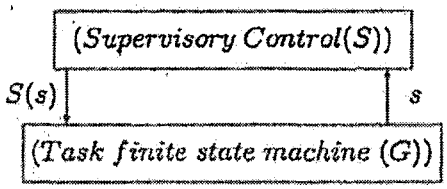

Figure 7.1: A Closed loop supervisory system

A supervisory policy and the associated state graph form a feedback control system. This closed loop supervisory system appears in figure 7.1. For every event produced by the state machine G ($s \in L(G)$) the supervisory control policy S has a set of enabled events ($S(s) \cap \Sigma_G(f(x_0, s))$) that G is allowed to execute from the current state $f(x_0, s)$. So, the supervisor may disable some of the events of G.

For complex graphs, investigation of supervisory policies that keep the desired safety properties and analysis of the properties of the closed loop systems formed by them would be useful extensions of this work.

Hidden Markov Modeling and Task Segmentation

Hidden Markov Models (HMM) and associated search algorithms can be used to integrate probabilistic determination of users intentions from the sensor input. Hannaford et al [Hannaford & Lee, 1991] among others have used HMMs extensively to model and analyze teleoperated tasks. For our purposes, a HMM is a finite state machine augmented with probability distributions for making transitions and observing state. The HMM could be initialized with the task graph. The probability distributions are initialized, with nominal values by observing the task. A set of trials are then performed for training the HMM. This trained HMM is then used for experimental purposes. This is a simple extension of our work. Pook [Pook, 1995] performed similar work for using a simple sign language to control a manipulator, and reports successful segmentation of user input.

Execution Environment

A simple interpreted execution environment was built for the purposes of our experiments. The data collected is processed offline, and is not used by the next execution of the same task graph. User profiles for task parameters is another useful feature currently not implemented. The execution environment can be extended to include support for process learning. It also needs better visualization of the data collected.

More Tasks

There is a wide range of applications for simple augmentation. We explored some applications in eye surgery. Similar applications in other surgical areas are also possible.

Stapedotomy

One avenue for application is ENT surgery. Excess bone tissue growth (Ostosclerosis) can cause the stapes bone in the ear to become attached to the surrounding bone of the middle ear. This restricts the movement of the stapes which impedes sound transmission causing deafness. Hearing can be restored using a procedure involving bypassing the stapes with a piston connected to the incus, and passing through a hole in the stapes. The procedure for drilling the hole is called stapedotomy. Since the thickness of the stapes varies, detecting the breakthrough is crucial in drilling or punching the right size hole.

Surgically augmenting this task involves constraining the motion of the robot appropriately to the location of the stapes, detecting contact, and punching or drilling until a breakthrough is detected. This is very similar to the vein cannulation task, the task graphs for this task will be very similar to vein cannulation.

An automatic system for stapedotomoy was reported by Brett [Brett *et al.*, 1995, Baker *et al.*, 1996]. Force profiles reported by Brett suggest the robust detection of breakthrough is possible. Rothbaum et al [Berkelman *et al.*, 2001] are independently exploring stapedotomy with the steady hand robot. Task level augmentation would likely increase transparency, safety, and improve performance.

Spinal surgery and pedicle screw placement

Spine surgery presents another application. Pedicle screws are a common means of addressing fractures and loss of strength in the spine. The incorrect placement of pedicle screws can cause damage to the pedicle wall as well as adjacent nerve roots. Several systems have been designed for monitoring the tools during pedicle screw

placement. There are no robotic systems for performing pedicle screw placement, although this is a topic of current research. Pedicle screw placement is a constrained motion task, involving registration of the robot to the patient, and constraining the robot to move along the axis of the hole to the specified depth. A task graph for this task would involve locating the robot at the position of the hole, and transitioning to a drilling action, that monitors the forces sensed, as well as verifies the hole using fluoroscopic imaging.

Other needle insertion tasks

Needle insertion tasks provide other opportunities. Prostate cancer treatment may involve delivering radiotherapy to selected portions of the prostate. Fine needle aspiration biopsy is often used to detect tuberculosis in the lungs. Jugular vein cannulation is commonly used to monitor blood flow. These, and other needle positioning tasks are also similar to the vein cannulation task, and could benefit from simple task augmentation.

Appendix A

A specification language

For describing simple manipulation tasks of interest using this paradigm a simple specification language was needed. Unlike a conventional programming language, most programs in this specification will construct and execute graphs on available or composed routines, instead of performing sequential computation.

A program consists of forward declaration of states of the graph (actions) and transitions (events), a task initialization section, definitions of the actions and events for each state(event map), and finally the events themselves.

```
action action1,action2;                    # forward declarations
event event0, event1, event2 ;
# task global initialization section
task a_task{     ...       }
# global event map - transitions active from all states
events {     ...       }
# task-state (action) definitions
action action1{     ...       }
# event map for this state
events {     ...       }
# definition of the transitions
event event-name {     ...       }
```

```
int i; double j;
frame f; rotation r;
vector v, v1;
i = 1; j = 2.5;
f = frame(45,0,0,0,1,2,3);
r = rotation(0, 45, 0, 0);
v = vector(1,1,1);
v1 = (!f)*v;
v1 = r*v1;
j = j*i;
```

Figure A.1: Possible declaration and usage of basic types. Cartesian frame f is constructed with a rotation of 45 degrees about x axis, and 0 degrees about y, and z, using a rotation order of X,Y,Z, and a vector (1,2,3). Rotation r is constructed to be a 45 degrees rotation about y axis. vector v1 gets the vector v transformed by the inverse of the frame f and is then rotated by rotation r.

A.1 Basic Data Types

The basic types needed include integers, double precision floating point numbers, and booleans. More important to manipulation and sensing programming are 2 and 3 dimensional vectors. Rotations and Cartesian frames are also essential basic types. Fixed length arrays are also supported.

The default origin for the frames is the fixed frame for the robot, and the rotation axes align with this coordinate frame as well. Mathematical operations on the basic types are defined. And most common algebraic and linear algebra operators are supported. Most commonly used mathematical and trigonometric functions are also supported.

A.2 Language Constructs

A variety of control structures are supported as statements. They include conditional and iterative execution constructs essential for any specification and constructs specific to building a task graph around the defined composite entities.

Variables form an essential part of any language. The basic types of the language were describe in the previous section and some examples present in figure A.1. The syntax of a variable declaration is:

```
typename var1 [,var2,..];
```

A variable can be initialized with a nominal value, expressions on the values of existing variables or values from the sensors. Nominal values, and expressions on other variables are assigned using the syntax:

```
varname = expression;
```

where as sensor values are acquired using:

```
acquire var;
```

Variables can also be read or written to the user interface. This input/output is performed by the syntax:

```
read var, [var1,..];
print var, [var1,..];
```

For accounting purposes, any variable, or expressions on variables can be monitored during the execution of a task. The syntax for recording a variable in the log files is:

```
log var, [var1,..];
```

Affixment

Often several variables track the value of a single sensor, or are defined with relation to each other. Affixment allows the value of a variable to be updated together with the value of the variable it is affixed to. The syntax for affixment is:

```
fix var1, expr; ;
Unfix var1, [var2..];
```

Affixed frames often track a single object. For example, for monitoring an offset frame from the tool, a frame variable can be affixed to the composition of the constant offset frame, and the tool frame. Any tool motion, results in the tool frame and consequently the affixed frame being updated automatically.

Conditional Execution Statements

The `if-statement` tests an expression and executes the `if` statement-list if the test is true, and the `else` statement-list otherwise. Chained `else-if` statements are also supported. The syntax is:

```
if (expression) statement-list
[else if (expression) statement-list]
[else statement-list];
```

Loop statements allow repeated execution of statement lists. The `for`, `while`, and -while constructs are supported. The syntax is:

```
for (assignment; expression; assignment)
    statement-list;
while(expression)
    statement-list;
do
    statement-list;
while (expression);
```

task graph construction

The construction of a task graph requires specification of its states (actions) and transitions (event maps), tests for the transitions (events), and initialization section for the task. The states and transitions of a task graphs are declared as forward

declarations. The syntax is:

```
action action-name, [action-name2,..];
event event-name, [event-name2,..];
```

To construct transitions, every action is provided with a list of events in its event map. The syntax for adding an event to the event map is:

```
On event-name action-name ;
```

Events return a value to invoke transitions. The return value is specified as:

```
return var;
```

Statement Lists

A statement-list consists of a list of statements enclosed in ({,}) parentheses. statement lists are used to create actions, events, event maps, and task initialization sections.

Composite Entities and Program Structure

The task initialization section describes the processing required before any manipulation can be performed. This includes setting up the workspace configuration and loading the required control parameters.

An action is a set of manipulation and sensing functions performing a part of the surgical task. An event evaluates expressions on sensed values to determine to control transition between actions.

An event map consists of on-statements on defined actions and events that constructs the possible transitions for an action.

All composite structures are specified as statement lists.

A.3 System Variables

The experimental system consists of the steady hand robot and additional sensors. A high resolution 3DOF tool tip force sensor is mounted before the tool mount. A

foot switch with 4 buttons and a 2DOF analog pedal joystick is available. 4 digital inputs and 4 digital outputs can be connected anywhere in the system. In addition, a line and a blob tracker are built into the system, if a calibrated camera is looking on configured objects, the position, size, and motion of 2 blobs, and a line can be monitored. All of these sensors are tracked as system variables.

A.4 Motion Routines

Cartesian level motions are issued by selecting a reference frame and providing a target frame to move. A move to a frame F is specified as follows:

```
targetframe= F;
MoveToFrame;
```

Utility cartesian level routines such as `MoveBaseToFrame`, `MoveArmToFrame` etc are also provided.

Joint level motions commands can be issued to move a set of joints to a particular position. A default list of joints contains all the joints of the robot. This can be reset to the required joints. Default lists for joint velocities and accelerations are also maintained and can be reset to required values. A simple motion for the first 5 joints to positions `[p0..p5]` could be specified as:

```
njoints=5;
jpos[0]=p1; ...  jpos[5]=p5;
PosMove;
```

Displacement motions can be specified by replacing (`PosMove`), with displacement (`PosDMove`) and velocity moves by specifying the velocities in `jvel` and using (`VelMove`). Accelerations are specified in `jaccel`. Since the robot is the only active device in the system all motion routines relate to the robot.

Values of all sensors are tracked as system variables. Routines are provided to resolve, bias, and limit these values.

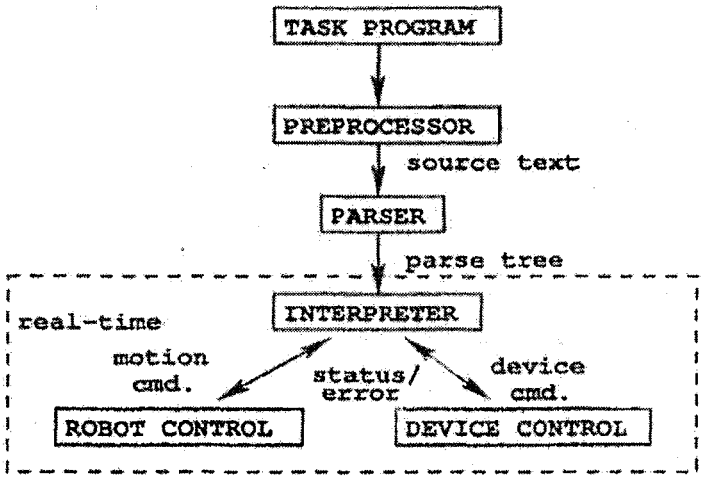

Figure A.2: The task program is preprocessed, and parsed to construct a parse tree. The interpreter uses the parse tree, and executes the task graph by calling appropriate robot and device control commands.

A.5 Run-Time Environment

A sketch of the run time system appears in figure A.2. The source task program is given to a preprocessor that checks the program structure and strips the source of comments. This reduced source is then parsed to construct a parse tree. The parse tree is The interpreter uses the parse tree, and executes the task graph by calling appropriate robot and device control commands.

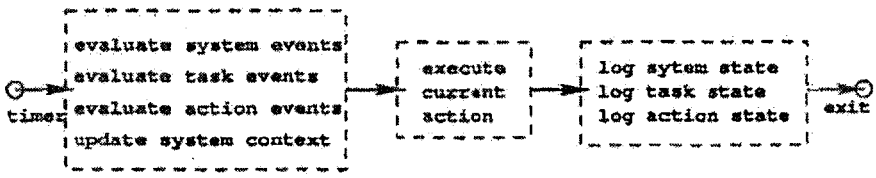

Figure A.3: Execution flow for the interpreter after initialization.

The Interpreter

An interpreter is used to execute these programs. A graphic user interface is provided to interact with the interpreter. An expression parser and evaluator is built into the debug system. Any sensing and manipulation routine can also be invoked in the interface. The main execution loop appears in figure A.3. A user interface thread is used to load programs and parse them. The initialization section (body of the task section) is then executed and control passed on to the first action.

Appendix B

Implementation of Task Graphs

This section details implementation of the task graphs used in the experiments section. This includes data collection for unassisted tasks, the peg in hole tasks, mosaicing tasks, and retinal vein cannulation tasks in chapter 5.

Recording unassisted performance

For the peg in hole experiments, and constrained peg in hole experiments simulating vein cannulation the unassisted mode experiments employed electric contact to detect a success or error. Each contact was counted once (so sliding along a surface was only counted once, but users were advised not to use strategies that included sliding). The system has four digital inputs `inp1` through `inp4` and four digital outputs `outp1` to `outp4`. For these experiments, the tool was connected to the first digital output, and the success and errors surfaces to the first two digital inputs. This simple task can be implemented with a single action that counts the successes and errors.

```
#----------------implementation of accounting task-----
action count;       #the only action
event event1;        # and an event allowing the user to quit
task account{        # task initialization section
   int success, error;
   bool intouch, intouch1;
```

```
    intouch=false;
    intouch1=false;
    success = 0;
    error = 0;
    outp1=true;
}
```

The variables success, error keep count of the numbers, where as intouch, intouch1 keep contact state. outp1, the first digital output line is attached to the tool, is set high during the initialization.

```
action count{
    if(inp2 && (!intouch1))
        { error = error + 1 ; intouch1=true;}
    else if(!inp2 ){intouch1=false;}
    if ( inp1 && (!intouch ) && (!inp2))
        { success = success + 1; intouch=true; }
    else if (!inp1){ intouch = false; }
    log time, success, error;
}
events { on event1 done;}
event event1{ return button1; }
```

The single action count checks the second digital input inp2, which is tied to the error surface and increments the number of errors if this is the first time an error has been noticed. Next, it checks the second digital input inp2, if this is the first time a contact has been noticed and there is no error contact, the number of successes is incremented.

Peg in hole experiments

The task graph used by assisted and augmented versions of the peg in hole experiments (figure 5.2) has robot modes very similar to the basic task graph in chapter 4 (figure 4.2). However, during different modes of the experiments the compliance gains for the robot are changed suitably.

The task has four actions. The actions `position` and `orient` are similar to actions `translate` and `orient` of the basic task graph. `Insert` and `retract` use gains that favor motion in the insertion direction. The task initialization section for the task is:

```
#------------peg in hole task---------------------
action position, orient, insert, retract;  #forward declarations
event event1,event2,event3;

task peginhole{# task init.  section
   flower=vector(-500,-500,-500);  # limits on the user forces
   fupper=vector(500,500,500);
   fconv=vector(2,2,2);            # scale user forces
   BiasUserForces;                 #compute bias forces:hands off
   Enable;                         #accept motion commands
}
```

Event1 monitors button1, event2 monitors button2, and event3 monitors button4 on the footpedal. The vector `fconv` is used to scale the user forces before they are used by the compliance routines. The scaling factor `fconv` is set to `vector(2,2,5)` for insertion and retraction to favor the direction of insertion. The action implementations are:

```
action position{
   fconv=vector(2,2,2);
   ComplyBaseJoints;
}
events {
   on event1 orient;
   on event2 insert;
}

action orient{
   fconv=vector(2,2,2);
   ComplyArmJoints;
}
events {
   on event2 position;
   on event1 insert;
}
```

```
action insert{
   fconv=vector(2,2,5);
   ComplyBaseJoints;
}
events {
   on event2 orient;
   on event1 retract;
}

action retract{
   fconv=vector(2,2,5);
   ComplyBaseJoints;
}
events {
   on event1 position;
   on event2 insert;
   on event3 done;
}
```

Creating retinal mosaics

The task graph for creating retinal mosaics (figure 5.7) has actions for positioning the endoscope where the images are to be taken, for taking images and moving the robot to the next position and finally retraction of the endoscope. The task initialization section is similar to the one above. The positioning and retraction actions are also similar. The actions increment and image implement a strategy for spiral or grid motion of the robot. For example, the strategy to image a grid region is implemented by the following code:

```
action increment{
    position=curframe;    #get the current position
```

```
    incremented=false;    #moved after next_step update
    # user can only move along x axis
    if((moveright (position<next_step))    #moving right
        ((!moveright)(position>next_step)))   # or moving left
          ComplyBaseJoints;
}
action image{
    if(!incremented){  #if motion since last update
        if(moveright){   #if moving right
          fconv=vector(2,0,0);   #allow motion only along x axis
          next_step=next_step+vector(increment,0,0);
        }
        else{
          fconv=vector(2,0,0);
          next_step=next_step-vector(increment,0,0);   #moving left
        }
        if( moveright  (next_step[0]>size[0])){#moving right
          moveright=false; #move left after this
          next_step=next_step+vector(0,increment,0); #move down once
          fconv=vector(0,2,0); #allow only up/down motion
        }
        if((!moveright)  (next_step[0]<start[0])){#moving left
          moveright=true; #move left after this
          next_step=next_step+vector(0,increment,0); #move down
          fconv=vector(0,2,0); #allow only up/down motion
        }
    }
}
```

The action increment allows the user to move the robot to the next position, and

stops when the position is reached. The user then presses the foot pedal button to invoke the image action, and an image is captured (the images here were captured by a separate computer). The image action increments next_step appropriately. The user can then choose to either end the program, or go back to increment and move the robot to the next position.

Retinal vein cannulation

The task initialization for vein cannulation (figure 5.13) is similar to earlier tasks. The task has seven task specific actions. These actions are implemented as:

```
action movetoport{
   fconv=vector(2,2,2);
   ComplyBaseJoints;
}
events {
   on event1 orient;
}

action orient{
   fconv=vector(2,2,2);
   ComplyArmJoints;
}
events {
   on event2 movetoport;
   on event1 insert;
}
```

```
action insert{
   fconv=vector(2,2,5);
   ComplyArmJoints;
}
events {
   on event2 orient;
   on contactevent contact;
}

action retract{
   fconv=vector(2,2,5);
   ComplyArmJoints;
}
events {
   on event1 done;
}
```

The actions hold, and contact are empty actions to allow the user to pause between contact and puncture and puncture and hold. The puncture action calls a puncture routine that uses the contact force profile to perform and automatic puncture.

Bibliography

[Baker *et al.*, 1996] Baker, D. A., Brett, P. N., Griffeth, M.V., & Reyes, L. 1996. Surgical Requirements for the stapedotomy tool: data and safety considerations. *Pages 214–215 of: 18th annual conference of the IEEE Engineering in Medicine and Biology Society.*

[Barnes, 1999] Barnes, A. C. 1999. *A Modular Robotic System For Precise Minimally Invasive Surgery.* M.Phil. thesis, The Johns Hopkins University.

[Berkelman *et al.*, 2000] Berkelman, P. J., Whitcomb, L. L., Taylor, R. H., & Jensen, P. S. 2000. A Miniature Instrument Tip Force Sensor for Robot/Human Cooperative Microsurgical Manipulation with Enhanced Force Feedback. *Pages 897–906 of: Medical Image Computing and Computer Assisted Surgery (MICCAI), Pittsburgh, PA, USA.* Lecture Notes In Computer Science, vol. 1935. Springer Verlag.

[Berkelman *et al.*, 2001] Berkelman, P.J., Rothbaum, D.L., Roy, J., Lang, S., Whitcomb, L.L., Hager, G., Jensen, P.S., deJuan, E., Taylor, R.H., & Niparko, J.K. 2001. Performance Evaluation of a Cooperative Manipulation Microsurgical Assistant Robot Applied to Stapedotomy. *Page submitted of: Medical Image Computing and Computer Assisted Surgery (MICCAI), Utrecht, Netherlands.*

[Bettini *et al.*, 2001] Bettini, A., Lang, S., Okamura, A., & Hager, G. 2001. Vision assisted control for manipulation using visual fixtures. *Page Submitted of: IEEE IROS.*

[Bolles & Paul, 1973] Bolles, R., & Paul, R. 1973. *The use of sensory feedback in a programmable assembly system.* Memo 220. Stanford AI Laboratory.

[Brett *et al.*, 1995] Brett, P. N., Baker, D. A., Reyes, L., & Blanshard, J. 1995. An automatic technique for micro-drilling a stapedotomy in the flexible stapes footplate. *Proc Instn Mech Engrs*, 209(H), 255–262.

[Brooks, 1985] Brooks, R. A. 1985. A Robust Layered Control System for a Mobile Robot. *IEEE Transactions on Robotics and Automation*, 2(1), 14–23.

[Brooks, 1990] Brooks, R. A. 1990. *The Behavior Language User's Guide.*

[Bzostek *et al.*, 2000] Bzostek, A., Kumar, R., Hata, N., Schorr, O., Kikinis, R., & Taylor, R.H. 2000. Distributed Modular Computer-Integrated Surgical Systems: Implementation using Modular Software and Networked Systems. *Pages 969–978 of:* Delp, S.L., DiGioia, A.M., & Jaramaz, B. (eds), *Medical Image Computing and Computer Assisted Surgery (MICCAI), Pittsburgh, PA, USA.* Lecture Notes In Computer Science, vol. 1935. Springer-Verlag.

[Cassandras *et al.*, 1995] Cassandras, C.G., Lafortune, S., & Olsder, G.J. 1995. Introduction to the modelling, control and optimization of discrete event systems. *Pages 217–291 of:* Isidori, A. (ed), *Trends in Control: A European Perspective.* Springer-Verlag.

[Charbonnier *et al.*, 1999] Charbonnier, F., Alla, H., & David, R. 1999. The Supervised Control of Discrete Event Dynamic Systems. *IEEE transactions on control systems technology*, 7(2), 175–187.

[Charles *et al.*, 1989] Charles, S., Williams, R. E., & Hamel, B. 1989. Design of a Surgeon-Machine Interface for Teleoperated Microsurgery. *Proc. of the Annual Int'l Conf. of the IEEE Engineering in Medicine and Biology Society*, 11, 883–884.

[Chen *et al.*, 2000] Chen, Y-L., Lafortune, S., & Lin, F. 2000. Design of Non-blocking

modular supervisors using event priority functions. *IEEE transactions on automatic control*, 45(3), 432–451.

[Cleeves & Findley, 1989] Cleeves, L., & Findley, L.J. 1989. Tremors. *Medical Clinics of North America*, 73(6), 1307–1319.

[Codourey *et al.*, 1997] Codourey, A., Rodriguez, M., & Pappas, I. 1997. A Task-oriented Teleoperation System for Assembly in the Microworld. *Pages 235–240 of: ICAR'97.*

[Das *et al.*, 1999] Das, H., Zak, H., Johnson, J., Crouch, J., & Frambach, D. 1999. Evaluation of a Telerobotic System to Assist Surgeons in Microsurgery. *Computer Assisted Surgery*, 4(15), 15–25.

[Davies *et al.*, 1997a] Davies, B. L., Fan, K. L., Hibberd, R. D., Jakopec, M., & Harris, S. J. 1997a. ACROBOT - Using Robots and Surgeons Synergistically in Knee Surgery. *In: 8th International Conference on Advanced Robotics.*

[Davies *et al.*, 1997b] Davies, B. L., Harris, S. J., Lin, W. J., Hibberd, R. D., Middleton, R., & Cobb, J. C. 1997b. Active Compliance in robotic surgery - the use of force control as a dynamic constraint. *Proc. Instn. Mech. Engrs.*, **211**(H), 285–292.

[Eberman, 1995] Eberman, B. 1995. *Contact Sensing: A Sequential Decision Approach to Sensing Manipulation Contact Features.* Ph.D. thesis, MIT.

[Finkel *et al.*, 1975] Finkel, R., Taylor, R.H., Bolles, R., Paul, R., & Feldman, J. 1975. AL, A programming system for automation. *Pages 758–765 of: Proceedings of the 4th international joint conference on artificial intelligence.*

[Gupta *et al.*, 1999] Gupta, P. K., Jensen, P.S., & de Juan, E. 1999. Surgical forces and tactile perception during retinal mi-crosurgery. *Pages 1218–1225 of:* Taylor, C, & Colchester, A (eds), *Medical Image Computing and Computer Assisted Surgery (MICCAI), Cambridge, UK.* Lecture Notes In Computer Science, vol. 1679. Springer-Verlag.

[Hannaford & Lee, 1991] Hannaford, B., & Lee, P. 1991. Hidden Markov Model analysis of force/torque information in telemanipulation. *International Journal of Robotics Research*, **10**(5), 528–539.

[Hogan, 1985] Hogan, N. 1985. Impedance control: An approach to manipulation. *ASME Journal of Dynamic Systems: Measurement and Control*, **107**, 1–7.

[Howe & Cutkosky, 1991] Howe, R. D., & Cutkosky, M.R. 1991. Touch sensing for robotic manipulation and recognition. *In:* Khatib, O., Craig, J., & Lozano-Pérez, T. (eds), *Robotics review 2.* MIT Press.

[Hunter *et al.*, 1993] Hunter, I. W., Jones, L.A., Sagar, M.A., Mallinson, G.D., & Hunter, P.J. 1993. A Teleoperated Microsurgical Robot and Assisted Virtual Environment for Eye Surgery. *Presence*, **2**(4), 265–280.

[Iba *et al.*, 1999] Iba, S., Weghe, M.V., Paredis, C.J.J., & Khosla, P.K. 1999. An architecture for gesture-based control of mobile robots. *Pages 851–857 of: IEEE International Conference on Intelligent Robots and Systems (IROS)*, vol. 2.

[Jensen *et al.*, 1997] Jensen, P. S., Grace, K. W., Attariwala, R., Colgate, J. E., & Glucksberg, M. R. 1997. Toward robot assisted vascular microsurgery in the retina. *Graefes Arch Clin Exp Ophthalmol*, **235**(11), 696–701.

[Kang & Ikeuchi, 1997] Kang, S. B., & Ikeuchi, K. 1997. Towards Automatic Robot Instruction from Perception: Mapping Human Grasps to Manipulator Grasps. *IEEE Transactions on Robotics and Automation*, **13**(1), 81–95.

[Kazerooni, 1989a] Kazerooni, H. 1989a. Human/robot interaction via the transfer of power and information signals - part i: Dynamics and control analysis. *Pages 1632–1640 of: IEEE International Conference on Robotics and Automation*, vol. 3.

[Kazerooni, 1989b] Kazerooni, H. 1989b. Human/robot interaction via the transfer of power and information signals - part ii: Dynamics and control analysis. *Pages 1641–1649 of: IEEE International Conference on Robotics and Automation*, vol. 3.

[Kazerooni & Jenhwa, 1993] Kazerooni, H., & Jenhwa, G. 1993. Human extenders. *ASME Journal of Dynamic Systems: Measurement and Control*, **115**(2B), 218–90.

[Kohner *et al.*, (1983] Kohner, E. M., Laatikainen, L., & Oughton, J. (1983. The management of retinal vein occlusion. *American Academy of Ophthalmology*, **83**, 484–487.

[Kosuge & Kazamura, 1997] Kosuge, K., & Kazamura, N. 1997. Control of a robot handling an object in cooperation with a human. *Pages 142–146 of: IEEE international Workshop on Robot Human Communication.*

[Kosuge *et al.*, 1993a] Kosuge, K, Fujisawa, Y., & Fukuda, T. 1993a. Control of man-machine system for dexterous manipulation. *Pages 249–256 of: Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2057.

[Kosuge *et al.*, 1993b] Kosuge, K., Fujisava, Y., & Fukuda, T. 1993b. Control of Robot Directly Maneuvered by Operator. *Pages 49–54 of: IEEE International Conference on Robots and Systems.*

[Kosuge *et al.*, 1993c] Kosuge, K., Yoshida, H., & Fukuda, T. 1993c. Dynamic Control for Robot-Human Collaboration. *Pages 398–401 of: IEEE International Workshop on Robot Human Cooperation.*

[Kumar *et al.*, 1999] Kumar, R., Goradia, T. M., Barnes, A. C., Jensen, P., Whitcomb, L. L., Stoianovici, D., Auer, L. M., & Taylor, R. H. 1999. Performance of robotic augmentation in microsurgery scale motions. *Pages 1108–1115 of:* Taylor, C, & Colchester, A (eds), *Medical Image Computing and Computer Assisted Surgery (MICCAI), Cambridge, UK.* Lecture Notes In Computer Science, vol. 1679. Springer-Verlag.

[Kumar *et al.*, 2000] Kumar, R., Hager, G.D., Barnes, A.C., Jensen, P.S., & Taylor, R.H. 2000. An Augmentation System for Fine Manipulation. *Pages 956–969 of:* Delp, S.L., DiGioia, A.M., & Jaramaz, B. (eds), *Medical Image Computing and*

Computer Assisted Surgery (MICCAI), Pittsburgh, PA, USA. Lecture Notes In Computer Science, vol. 1935. Springer-Verlag.

[Lozano-Pérez, 1976] Lozano-Pérez, T. 1976. *The design of a mechanical assembly system.* M.Phil. thesis, Massachustetts Institute of Technology.

[Lozano-Pérez *et al.*, 1984] Lozano-Pérez, T, Mason, M.T, & Taylor, R.H. 1984. Automatic Synthesis of Fine Motion Strategies for Robots. *International Journal of Robotics Research*, 3(1), 3–24.

[Mason, 1978] Mason, M. T. 1978. *Compliance and force control for computer controlled manipulators.* M.Phil. thesis, MIT, Cambridge, MA.

[Mason, 1982] Mason, M. T. 1982. Compliance and force control for computer controlled manipulators. *Pages 373–404 of:* Brady, B., Hollerbach, J. M., Johnson, T. L., Lozano-Pérez, T., & Mason, M. T. (eds), *Robot Motion: Planning and Control.* Cambridge, MA: MIT Press.

[Misuishi *et al.*, 1997] Misuishi, M., Watanabe, H., Nakanishi, H., Kubota, H., & IIzuka, Y. 1997. Dexterity enhancement for a tele-micro-surgery system with multiple macro-micro co-located operation point manipulators and understanding of the operator's intention. *Pages 821–830 of:* Troccaz, J., Grimson, E., & Mosges, R. (eds), *First joint conference computer vision, virtual realtiy and robotics in medicine and medical robotics and computer-assisted surgery.* Grenoble, France: Springer.

[Morrow & Khosla, 1997] Morrow, J. D., & Khosla, P. K. 1997. Manipulation Task Primitives for Composing Robot Skills. *Pages 3354–3359 of: IEEE International Conference on Robotics and Automation*, vol. 4.

[Muller *et al.*, 1996] Muller, L., Hecht, M. H., Miller, L. M., Rockstad, H. K., & Lyke, J. C. 1996. Packaging and qualification of MEMS-based space systems. *Pages 503–508 of: Proceedings of Workshop on MEMS.*

[Paul *et al.*, 1992] Paul, H., Mittelstadt, B., Bargar, W., Musits, B., Taylor, R. H., Kazanzides, P., Zuhars, J., Williamson, B., & Hanson, W. 1992. A Surgical Robot for Total Hip Replacement Surgery. *In: IEEE International Conference on Robotics and Automation.* Nice, France: IEEE Press.

[Pook, 1995] Pook, P. K. 1995. *Teleassistance: Using Deictic Gestures to Control Robot Motion.* Ph.D. Thesis, University of Rochester.

[Ramadge & Wonham, 1989] Ramadge, P.J., & Wonham, W.M. 1989. The Control of Descrete Event Systems. *Proceedings of the IEEE*, **79**(1), 81–98.

[Roy & Whitcomb, 2001] Roy, J., & Whitcomb, L.L. 2001. Adaptive Force Control of Position/Velocity Controlled Robot Arms: Theory and Experiments. *IEEE transactions on Robotics and Automation.*

[Salcudean *et al.*, 1997] Salcudean, S. E., Ku, S., & Bell, G. 1997. Performance measurement in scaled teleoperation for microsurgery. *Pages 789–798 of:* Troccaz, J., Grimson, E., & Mosges, R. (eds), *First joint conference computer vision, virtual realtiy and robotics in medicine and medical robotics and computer-assisted surgery.* Grenoble, France: Springer.

[Salisbury, 1980] Salisbury, J. K. 1980. Active Stiffness Control of a Manipulator in Cartesian Coordinates. *Pages 95–100 of: 19th IEEE Conference on Decision and Control.*

[Sanderson & Perry, 1983] Sanderson, A C, & Perry, G. 1983. Sensor Based Robotic Assembly Systems: Research And Applications in Electronic Manufacturing. *Proceedings of the IEEE*, **71**(7), 856–871.

[Sanderson *et al.*, 1990] Sanderson, A.C., de Mello, L. S. H., & Zhang, H. 1990. Assembly sequence planning. *AI Magazine*, **11**(1), 62–81.

[Schenker *et al.*, 1995] Schenker, P.S., Das, H.O., & Timothy, R. 1995. Development of a new high-dexterity manipulator for robot-assisted microsurgery. *Pages 191–*

198 of: Proceedings of SPIE - The International Society for Optical Engineering: Telemanipulator and Telepresence Technologies, vol. 2351.

[Sheridan, 1988] Sheridan, T.B. 1988. Task Allocation and Supervisory Control. *Pages 159–173 of:* Helander, M. (ed), *Handbook of Human Computer Interaction.* Elsevier Science Publishers B.V.

[Sheridan, 1998] Sheridan, T.B. 1998. Human Factors in Tele-inspection and Telesurgery:Cooperative manipulation under Asynchronous Video and Control Feedback. *Pages 368–376 of:* Wells, W, Colchester, A, & Delp, S (eds), *Medical Image Computing and Computer Assisted Surgery (MICCAI), Cambridge, MA, USA.* Lecture notes in computer science, vol. 1496. Springer-Verlag.

[Stoianovici *et al.*, 1998] Stoianovici, D., Whitcomb, L. L., Anderson, J. H., Taylor, R. H., & Kavoussi, L. R. 1998. A Modular Surgical Robotic System for Image Guided Percutaneous Procedures. *Pages 404–410 of:* Wells, W, Colchester, A, & Delp, S (eds), *Medical Image Computing and Computer Assisted Surgery (MICCAI), Cambridge, MA, USA.* Lecture notes in computer science, vol. 1496. Cambridge, USA: Springer-Verlag.

[Taylor, 1976] Taylor, R. H. 1976. *The Synthesis of Manipulator Control Programs from Task Level Specifications.* Ph.D. thesis, Stanford University.

[Taylor, 1999] Taylor, R. H. 1999. Medical Robotics. *Pages 1213–1230 of:* Nof, Shimon Y. (ed), *Handbook of Industrial Robotics, Second Edition.* New York: Wiley.

[Taylor *et al.*, 1985] Taylor, R. H., Korein, J., Maier, G., & Durfee, L. 1985. A General Purpose Architecture for Programmable Automation Research. *In: Third Int. Symposium on Robotics Research.* Chantilly: MIT Press.

[Taylor *et al.*, 1994] Taylor, R. H., Paul, H. A., Kazandzides, P., Mittelstadt, B. D., Hanson, W., Zuhars, J. F., Williamson, B., Musits, B. L., Glassman, E., & Bargar, W. L. 1994. An Image-directed Robotic System for Precise Orthopaedic Surgery. *IEEE Transactions on Robotics and Automation*, 10(3), 261–275.

[Taylor *et al.*, 1996] Taylor, R. H., Funda, J., Eldridge, B., Gruben, K., LaRose, D., Gomory, S., & Talamini, M. 1996. A Telerobotic Assistant for Laparoscopic Surgery. *Pages 581–592 of:* Taylor, R., Lavallee, S., Burdea, G., & Moesges, R. (eds), *Computer-Integrated Surgery.* MIT Press.

[Taylor *et al.*, 1999a] Taylor, R. H., Jensen, P., Whitcomb, L. L., Barnes, A., Kumar, R., Stoianovici, D., Gupta, P., Wang, Z., deJuan, E., & Kavoussi, L. 1999a. A Steady-Hand Robotic System for Microsurgical Augmentation. *Pages 1031–1040 of:* Taylor, C, & Colchester, A (eds), *Medical Image Computing and Computer Assisted Surgery (MICCAI), Cambridge, UK.* Lecture notes in computer science, vol. 1679. Springer Verlag.

[Taylor *et al.*, 1999b] Taylor, R. H., Jensen, P., Whitcomb, L. L., Barnes, A., Kumar, R., Stoianovici, D., Gupta, P., Wang, Z., deJuan, E., & Kavoussi, L. 1999b. A Steady-Hand Robotic System for Microsurgical Augmentation. *International Journal of Robotics Research*, **18**(12), 1201–1210.

[Taylor *et al.*, 1982] Taylor, R.H., Summers, P., & Meyer, J. 1982. AML : A manufacturing Language. *International Journal of Robotics Research*, 1(3), 19–41.

[Troccaz & Delnondedieu, 1996] Troccaz, J., & Delnondedieu, Y. 1996. Semi-active guiding systems in surgery. A two dof prototype of the passive arm with dynamic constraints(PADyC). *Mechatronics*, 6(4), 399–421.

[Whitney, 1977] Whitney, D. E. 1977. Force Feedback Control of Manipulator Fine Motions. *ASME Journal of Dynamic Systems: Measurement and Control*, 91–97.

[Woo *et al.*, 1998] Woo, K. Y., Jin, B. D., & Kwon, D-S. 1998. A 6DOF Force-Reflecting Hand Controller Using Fivebar parallel Mechanism. *Pages 1597–1602 of: IEEE International Conference on Robotics and Automation.*

Vita

Rajesh Kumar was born August 8, 1973 in Haryana, India. He attended Delhi Institute of Technology, University of Delhi, India graduating with the Bachelor of Engineering degree in Computer Engineering in 1994. After spending a year and half at the Center of Development of Telematics (C-DOT), a government of India telecommunications research center, he began graduate studies at The Johns Hopkins Univeristy in 1996. At Hopkins, Kumar pursued his research in application of computing and robotics to surgery, particularly in development of new systems for computer integrated surgery, designing and implementing modular software architectures for the same, investigating novel robot assisted procedures, and evaluating human performance using these systems, primarily under the guidance of Dr. Russell H. Taylor.